

IDC DOCUMENTATION

Surface Wave Identification and Measurement



Notice

This document was published March 2001 by the Monitoring Systems Operation of Science Applications International Corporation (SAIC) as part of the International Data Centre (IDC) Documentation. Every effort was made to ensure that the information in this document was accurate at the time of publication. However, information is subject to change.

Contributors

Jeffrey L. Stevens, Maxwell Technologies

Hans Israelsson, Science Applications International Corporation

Trademarks

BEA TUXEDO is a registered trademark of BEA Systems, Inc.

ORACLE is a registered trademark of Oracle Corporation.

SAIC is a trademark of Science Applications International Corporation.

Solaris is a registered trademark of Sun Microsystems.

SPARC is a registered trademark of Sun Microsystems.

SQL*Plus is a registered trademark of Oracle Corporation.

Sun is a registered trademark of Sun Microsystems.

UltraSPARC is a registered trademark of Sun Microsystems.

UNIX is a registered trademark of UNIX System Labs, Inc.

Ordering Information

The ordering number for this document is SAIC-01/3008.

This document is cited within other IDC documents as [IDC7.1.3].

Surface Wave Identification and Measurement

CONTENTS

About this Document	i
■ PURPOSE	ii
■ SCOPE	ii
■ AUDIENCE	ii
■ RELATED INFORMATION	iii
■ USING THIS DOCUMENT	iii
Conventions	iv
Chapter 1: Overview	1
■ INTRODUCTION	2
■ FUNCTIONALITY	7
■ IDENTIFICATION	7
■ STATUS OF DEVELOPMENT	8
■ BACKGROUND AND HISTORY	8
■ OPERATING ENVIRONMENT	9
Hardware	9
Commercial-off-the-shelf Software	10
Chapter 2: Architectural Design	11
■ CONCEPTUAL DESIGN	12
■ DESIGN DECISIONS	12
Programming Language	13
Global Libraries	13
Database	14
File System	14
Design Model	14
Database Schema Overview	15

■ FUNCTIONAL DESCRIPTION	16
Creating Binary Files from Regionalized Dispersion Curves	18
Determining Origins for Processing	18
Identifying and Measuring Surface Waves	18
Calculating Magnitudes	19
Identifying and Removing Spurious and Misassociated Arrivals	19
■ INTERFACE DESIGN	20
Interface with Other IDC Systems	21
Interface with Operators	21
Chapter 3: Detailed Design	23
■ DATA FLOW MODEL	24
■ PROCESSING UNITS	27
LPcompile	28
MsInterval	32
MsOrid	36
maxsurf	37
MsConflict	61
libLP	62
liblpararray	63
liblppar	64
libmag	66
libnbf	68
libpathcor	69
libpmotion	70
libresp	71
libs3obj	73
libutilc	74
libutilf	74
■ DATABASE DESCRIPTION	75
Database Design	75
Database Schema	76
References	81

Glossary

G1

Index

I1

Surface Wave Identification and Measurement

FIGURES

FIGURE 1.	IDC SOFTWARE CONFIGURATION HIERARCHY	4
FIGURE 2.	RELATIONSHIP AMONG AUTOMATIC PROCESSING CSCI SOFTWARE AND OTHER CSCIs	6
FIGURE 3.	REPRESENTATIVE HARDWARE CONFIGURATION FOR SWIM SOFTWARE	10
FIGURE 4.	PROCESSING FLOW OF SWIM SOFTWARE	17
FIGURE 5.	TABLE RELATIONSHIPS FOR SWIM SOFTWARE	80

Surface Wave Identification and Measurement

TABLES

TABLE I:	DATA FLOW SYMBOLS	iv
TABLE II:	ENTITY-RELATIONSHIP SYMBOLS	v
TABLE III:	TYPOGRAPHICAL CONVENTIONS	v
TABLE 1:	SWIM SOFTWARE PROGRAMMING LANGUAGES	13
TABLE 2:	DATABASE TABLES USED BY SWIM SOFTWARE	15
TABLE 3:	DATABASE USAGE BY SWIM SOFTWARE	76

About this Document

This chapter describes the organization and content of the document and includes the following topics:

- Purpose
- Scope
- Audience
- Related Information
- Using this Document

About this Document

PURPOSE

This document describes the design of the surface wave identification and measurement (SWIM) software of the International Data Centre (IDC). The software is a computer software component (CSC) of the Automatic Processing Computer Software Configuration Item (CSCI). This document provides a basis for implementing, supporting, and testing the software.

SCOPE

The suite of software for identifying and measuring surface waves is identified as follows:

Title: surface wave identification and measurement

Abbreviation: SWIM

This document describes the architectural and detailed design of the software including its functionality, components, data structures, high-level interfaces, method of execution, and underlying hardware. This information is modeled on the Data Item Description for Software Design Descriptions [DOD94a].

AUDIENCE

This document is intended for all engineering and management staff concerned with the design of all IDC software in general and of the SWIM software in particular. The detailed descriptions are intended for programmers who will be developing, testing, or maintaining the software.

RELATED INFORMATION

The following documents complement this document:

- *IDC Processing of Seismic, Hydroacoustic, and Infrasonic Data* [IDC5.2.1]
- *Regionalized Maximum Likelihood Surface Wave Analysis* [Ste96]
- *Improved Methods for Regionalized Surface Wave Analysis* [Ste97]
- *Optimization of Surface Wave Identification and Measurement* [Ste01]

See “References” on page 81 for a list of documents that supplement this document. The following UNIX man pages apply to the existing SWIM software:

- *maxsurf*
- *LPcompile*

USING THIS DOCUMENT

This document is part of the overall documentation architecture for the IDC. It is part of the Software category, which describes the design of the software. This document is organized as follows:

- Chapter 1: Overview
This chapter provides a high-level view of the SWIM software, including its functionality, components, background, status of development, and current operating environment.
- Chapter 2: Architectural Design
This chapter describes the architectural design of the SWIM software, including its conceptual design, design decisions, functions, and interface design.
- Chapter 3: Detailed Design
This chapter describes the detailed design of SWIM software including its data flow, software units, and database design.
- References
This section lists the sources cited in this document.

▼ About this Document

- Glossary

This section defines the terms, abbreviations, and acronyms used in this document.

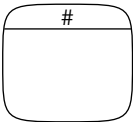


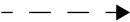

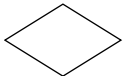
- Index

This section lists topics and features provided in the document along with page numbers for reference.

Conventions

This document uses a variety of conventions, which are described in the following tables. Table I shows the conventions for data flow diagrams. Table II shows the conventions for entity-relationship diagrams. Table III lists typographical conventions.

TABLE I: DATA FLOW SYMBOLS

Description ¹	Symbol
process	
external source or sink of data	
data store D = disk store Db = database store	
control flow	
data flow	
decision	

1. Symbols in this table are based on Gane-Sarson conventions.

TABLE II: ENTITY-RELATIONSHIP SYMBOLS




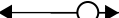






Description	Symbol			
One A maps to one B.	A  B			
One A maps to zero or one B.	A  B			
One A maps to many Bs.	A  B			
One A maps to zero or many Bs.	A  B			
database table	<table><tr><td>tablename</td></tr><tr><td> <i>primary key</i>  <i>foreign key</i></td></tr><tr><td><i>attribute 1</i> <i>attribute 2</i> ... <i>attribute n</i></td></tr></table>	tablename	 <i>primary key</i>  <i>foreign key</i>	<i>attribute 1</i> <i>attribute 2</i> ... <i>attribute n</i>
tablename				
 <i>primary key</i>  <i>foreign key</i>				
<i>attribute 1</i> <i>attribute 2</i> ... <i>attribute n</i>				

TABLE III: TYPOGRAPHICAL CONVENTIONS

Element	Font	Example
database table	bold	amplitude
database table and attribute when written in the dot notation		stamag.sta
database attributes	<i>italics</i>	<i>sta</i>
processes, software units, and libraries		<i>maxsurf</i>
user-defined arguments and variables used in parameter (par) files or program command lines		LPcompile LPdir
titles of documents		<i>Database Schema</i>
text that should be typed in exactly as shown	courier	LPcompile
default parameter values		Default: 0.2

Chapter 1: Overview

This chapter provides a general overview of the SWIM software and includes the following topics:

- Introduction
- Functionality
- Identification
- Status of Development
- Background and History
- Operating Environment

Chapter 1: Overview

INTRODUCTION

The software of the IDC acquires time-series and radionuclide data from stations of the International Monitoring System (IMS) and other locations. These data are passed through a number of automatic and interactive analysis stages, which culminate in the estimation of location and in the origin time of events (earthquakes, volcanic eruptions, and so on) in the earth, including its oceans and atmosphere. The results of the analysis are distributed to States Parties and other users by various means. Approximately one million lines of developmental software are spread across six CSCIs of the software architecture. One additional CSCI is devoted to run-time data of the software. Figure 1 shows the logical organization of the IDC software. The SWIM software is in the Automatic Processing CSCI as part of the Post-location Processing CSC (which is “boxed” in the figure). The Automatic Processing CSCI processes data through the following CSCs:

- Station Processing
This software scans data from individual time-series stations for characteristic changes in the waveforms (detection of onsets) and characterizes such onsets (feature extraction). The software then classifies the detections as arrivals in terms of phase type.
- Network Processing
This software combines arrivals from several stations originating from one event and infers the location and time of its origin.
- Post-location Processing
This software computes various magnitude estimates and selects data to be retrieved from auxiliary stations.

- Event Screening

This software extracts a number of parameters that characterize an event; then a default subset of the calculated event characterization parameters eliminates the events that are clearly not explosions.

- Time-series Tools

This software includes various utilities for the Seismic, Hydroacoustic, and Infrasonic (S/H/I) processing system.

- Time-series Libraries

This software includes shared libraries to which several modules of the S/H/I processing system are linked.

- Operational Scripts

This software provides miscellaneous functionality to enable automatic processing to function as a system.

- Radionuclide Processing

This software includes the automated analysis, categorization, and flagging processes for radionuclide data.

- Atmospheric Transport

This software includes the forward and backward modeling of the transport of particulates by atmospheric movements.

▼ Overview

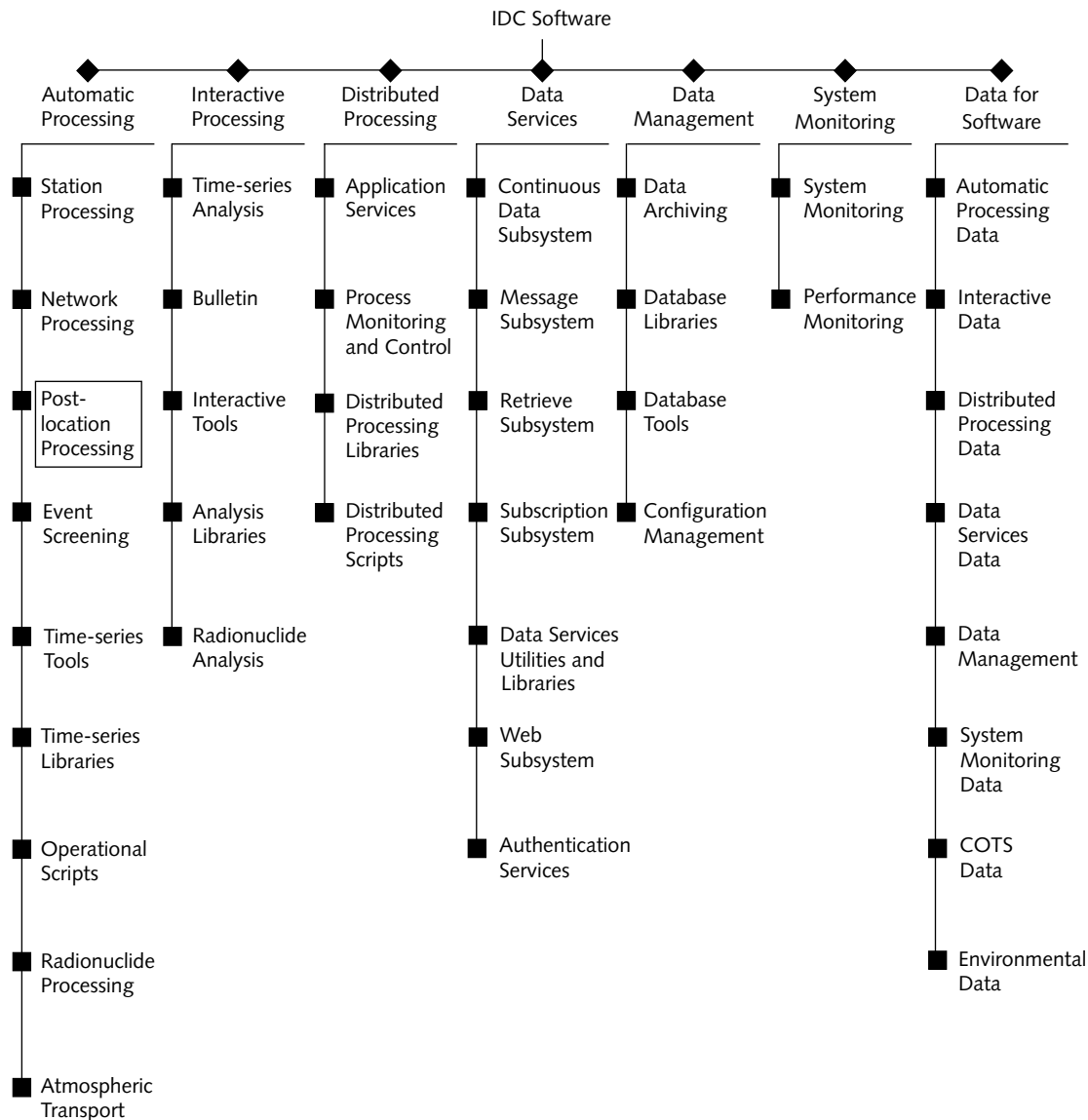


FIGURE 1. IDC SOFTWARE CONFIGURATION HIERARCHY

Figure 2 shows the relationship of the SWIM software to other components of the Automatic Processing CSCI and other CSCIs. Automatic identification and measurement of surface waves is implemented by the SWIM software, which runs during post-location processing, retrieving the event location and waveform information from the database. The main processing component of the SWIM software, *maxsurf*, writes results to the database, which are then used by the Event Location (*EvLoc*) software to calculate surface wave magnitudes.

▼ Overview

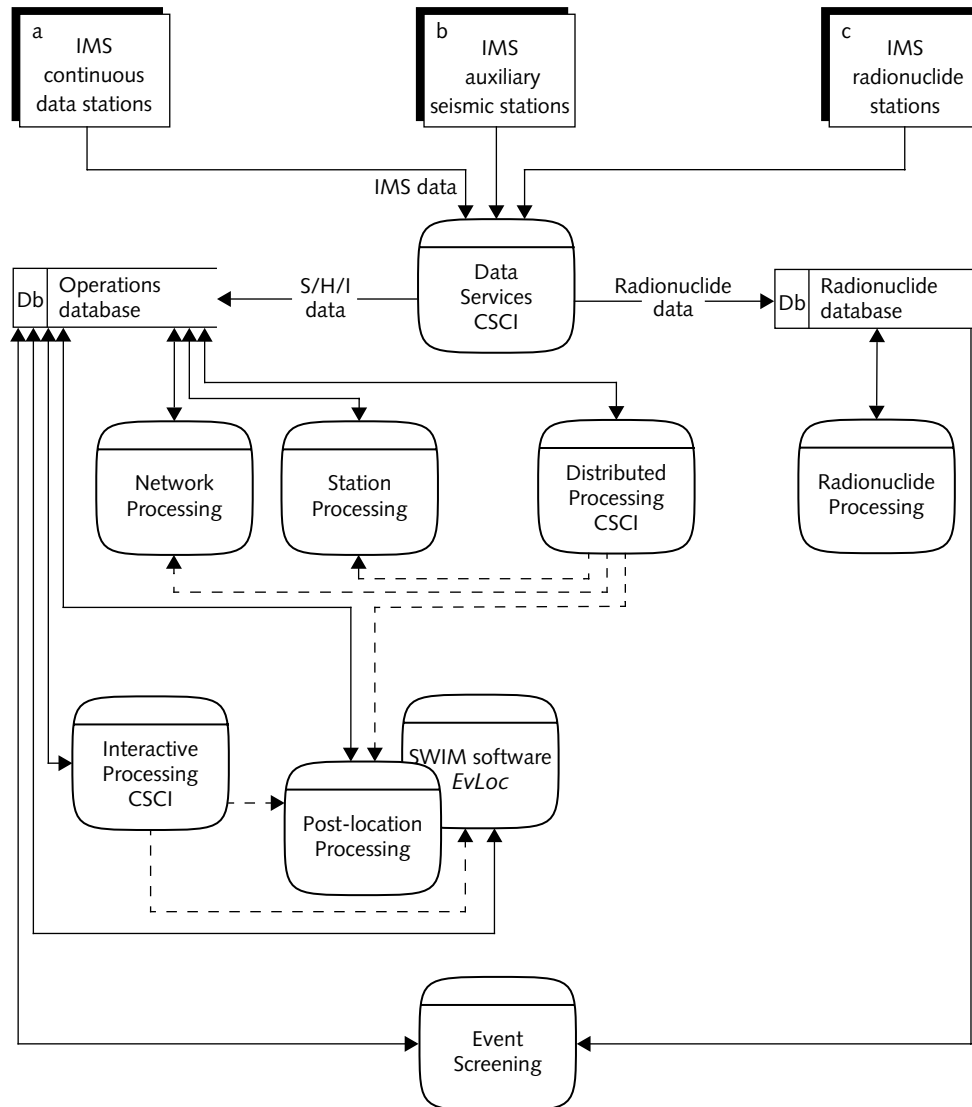


FIGURE 2. RELATIONSHIP AMONG AUTOMATIC PROCESSING CSCI SOFTWARE AND OTHER CSIS

FUNCTIONALITY

The SWIM software runs in the processing stream after events have been defined and located. Surface wave processing is applied to all continuous long-period and broadband data that are received by the IDC. The *maxsurf* program examines the arrival window where a surface wave would be expected and applies a dispersion test to see if a surface wave can be identified. If identified, the amplitude, period, arrival time, azimuth, and (if possible) slowness of the surface wave are measured and stored in the database. The *maxsurf* program calculates the surface wave magnitude M_s but does not store it in the database. The program *EvLoc* calculates M_s for all arrivals and stores the results in the database.

IDENTIFICATION

The SWIM software components are identified as follows:

- *LPcompile*
- *MsInterval*
- *MsOrid*
- *maxsurf*
- *MsConflict*
- *libLP*
- *liblpararray*
- *liblppar*
- *libmag*
- *libnbf*
- *libpathcor*
- *libpmotion*
- *libresp*
- *libs3obj*
- *linutilc*
- *libutilf*

STATUS OF DEVELOPMENT

The SWIM software was first implemented at the Prototype International Data Centre (PIDC) in May 1995. Several revisions have been made to the code and to the procedures used since that time, and the software was included in the first delivery of software to the permanent IDC in Vienna. The current version of the main SWIM software program (*maxsurf* 3.10) has been operational at the PIDC since 15 February, 1997. Improved dispersion curves were installed in May 1998. The distance correction used to calculate M_s was changed to that of Rezapour and Pearce [Rez98] in September 1998 (*maxsurf* and *EvLoc* use the same formula).

BACKGROUND AND HISTORY

Jeffrey L. Stevens and coworkers at Maxwell Technologies developed the first version of the SWIM software in 1995 and have continued to maintain and update the software since that time.

The SWIM software was first used operationally in May 1995 and is currently an element of the PIDC at the Center for Monitoring Research (CMR) in Arlington, Virginia, U.S.A., and the International Data Centre of the Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO IDC) in Vienna, Austria.

The SWIM software program *maxsurf* identifies surface waves by looking for characteristics of surface waves in the time interval where surface waves would be expected. The identification procedure has been improved substantially since *maxsurf* was first installed in 1995. The first version of *maxsurf* applied two tests: a simple dispersion test that looked for group velocity that decreased with increasing frequency and an azimuth test that compared the azimuth obtained from either an array beam or three-component (3-C) polarization filtering with the known azimuth. These early identification tests have now been replaced by comparison of measured dispersion with a regionalized global dispersion model and postprocessing to remove any remaining spurious or misassociated arrivals. The measured azimuths are still recorded in the database for each associated LR detection.

Surface wave measurement procedures have also evolved over time. The first step in measurement of a well-dispersed wavetrain is to decide which part of the wavetrain to measure. Prior to February 15, 1997, the measurement algorithm picked a peak close to 20 seconds using a weighting scheme based on the difference between the measured peak and 20 seconds. This caused a small bias to occur between PIDC measurements and United States Geological Survey (USGS) measurements, so the procedures were revised and are now identical to those used by the USGS, where the largest peak in the 18–22 second period range is used. To ensure consistency in measurement, seismograms are first transformed to a common (KS36000 long period) instrument. The largest peak-to-peak instrument corrected amplitude in the 18–22 second period range within the predicted arrival time window is then identified and measured and the LR amplitude, which is defined as half the peak-to-peak measurement, is stored in the **amplitude** and **arrival** database tables. At 3-C stations, the amplitude measurement is made on the vertical component, and at arrays it is made on a beam of the vertical components where the beam is constructed using the known azimuth to the event.

OPERATING ENVIRONMENT

The following paragraphs describe the hardware and commercial-off-the-shelf (COTS) software required to operate the SWIM software.

Hardware

The SWIM software is designed to run on a UNIX workstation such as the Sun UltraSPARC 5. Typically, the hardware is configured with 128 MB of memory and a minimum of 4 GB of magnetic disk. The compiled software modules, *maxsurf* and *LPcompile*, and associated scripts require about a megabyte of disk space. Regionalized dispersion curves require about 5 MB of disk space. The SWIM software writes diagnostic information to standard output. If the information is saved, the disk space used will scale with the number of stations in the network and the number of events processed (about 50 KB per event are produced with the current IMS network). The SWIM software must access database services over the local area network. Figure 3 shows a representative hardware configuration.

▼ Overview

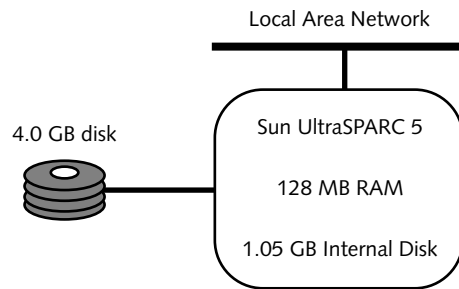


FIGURE 3. REPRESENTATIVE HARDWARE CONFIGURATION FOR SWIM SOFTWARE

Commercial-off-the-shelf Software

The SWIM software is designed for the Solaris 7.0 operating system and ORACLE 8.1.5 running on Sun workstations and servers. Distributed processing is controlled using BEA TUXEDO software.

Chapter 2: Architectural Design

This chapter describes the architectural design of the SWIM software and includes the following topics:

- Conceptual Design
- Design Decisions
- Functional Description
- Interface Design

Chapter 2: Architectural Design

CONCEPTUAL DESIGN

The SWIM software must identify and measure surface waves so that surface wave magnitudes, M_s , can be estimated. Surface waves are easier to identify after the event location and origin time are known, so surface wave processing is performed after locations have been determined based on body wave arrivals. Information about the origins is obtained from the database, the data are processed to detect and measure surface waves, surface wave magnitudes are determined, and the results are written to the database.

To process the data, the SWIM software must interact with the operational setup of the system in which it is installed. At the IDC, surface wave processing is performed during the analyst review process, mostly under the control of the Distributed Application Control System (DACS) (see [IDC6.5.2Rev0.1]). The software is thus designed to process data in time intervals that are provided as input.

The software first identifies events to process. A script, *MsInterval*, gets the event origins to be processed in a given time interval. *MsInterval* then calls another script, *MsOrid*, for each origin within the interval. *MsOrid* gathers information about the origin from the database, determines if the origin should be processed, and passes the information to the main processing routine, *maxsurf*. *MsOrid* then calls the *EvLoc* application, which estimates M_s and stores the results in the database (*EvLoc* is not covered by this document). Finally, another script, *MsConflict*, performs quality control of the results generated by *MsInterval* and *MsOrid*, removes spurious or misassociated arrivals from the database, and calls *EvLoc* to calculate the final M_s .

DESIGN DECISIONS

The following design decisions pertain to the SWIM software.

Programming Language

The SWIM software consists of two C programs (*maxsurf* and *LPcompile*), several libraries, and three Perl scripts (*MsInterval*, *MsOrigin*, and *MsConflict*).

TABLE 1: SWIM SOFTWARE PROGRAMMING LANGUAGES

Program or Library	Programming Language
<i>MsInterval</i>	Perl script
<i>MsOrid</i>	Perl script
<i>maxsurf</i>	C program
<i>MsConflict</i>	Perl script
<i>libLP</i>	C library
<i>liblpparray</i>	FORTTRAN and C library
<i>liblppar</i>	C library
<i>libmag</i>	FORTTRAN and C library
<i>libnbf</i>	FORTTRAN and C library
<i>libpathcor</i>	FORTTRAN and C library
<i>liblpmotion</i>	FORTTRAN library
<i>libresp</i>	C library
<i>libs3obj</i>	FORTTRAN and C library
<i>libutilc</i>	C library
<i>libutilf</i>	FORTTRAN library
<i>LPcompile</i>	C program

Global Libraries

The *maxsurf* program is linked to the following shared libraries: *libgeog*, *libresponse*, *libwfm*, *libgdi*, *libdb30aqa*, *libaesir*, and *libpar*.

▼ Architectural Design

The *maxsurf* program is also linked to the following COTS libraries: *libF77*, *libsocket*, *libnsl*, *libelf*, *libm*, and *libdl*. Because *maxsurf* uses *libgdi* for database access it is not necessary to explicitly link with ORACLE libraries. Linking is performed with Fortran 77 (F77).

Database

The SWIM software retrieves waveform, event, station, and instrument information from an ORACLE relational database management system (RDBMS) and writes arrival information back to the RDBMS. The database is also used by the DACS to manage the intervals processed.

File System

The file system is used to store the following files: program, scripts, parameter files, regionalized dispersion curves, instrument response files, and waveform data. On each run a small amount of diagnostic information is written to standard output and may be stored on the file system. All other input and output is stored in the database.

Design Model

The design of the SWIM software is primarily influenced by the need to identify and measure surface waves automatically. One consequence of this is that the SWIM software must extract information from the database that either changes frequently (for example, event location) or may change over time (for example, stations in the IMS network and instrument responses). The SWIM software results must be written to the database for use by other software modules and for bulletin generation. Another consequence of automatic processing is the need to coordinate the processing of different time intervals and to recover from failures. This aspect of the design model is handled by the DACS and is not discussed in this document.

Database Schema Overview

The SWIM software uses the ORACLE database for the following purposes:

- to retrieve information about the event being processed and the associated waveform segments
- to retrieve information about the IMS network, station locations, instrument orientations, and instrument responses
- to store processing results including amplitudes, periods, arrival times, and azimuth estimates that will be used by *EvLoc* to estimate surface wave magnitudes

Table 2 shows the database tables used by the SWIM software. The Name column identifies the database table. The Mode column is “R” if the SWIM software reads from the table and “W” if it writes to the table. The “DBA” (database administrator) has administrative control over all tables used by the SWIM software.

TABLE 2: DATABASE TABLES USED BY SWIM SOFTWARE

Name	Mode	Description
affiliation	R	station affiliations
amplitude	R/W	measured amplitudes for signal or noise
arrival	R/W	signal amplitude and arrival time
assoc	R/W	link between origin and arrival
instrument	R	pointer to instrument response files
netmag	R/W	network magnitude created by <i>EvLoc</i> , not <i>maxsurf</i> deleted if it exists by <i>MsOrid</i>
origin	R/W	event location and time information
origerr	R	origin error information
parrival	R/W	predicted arrival time for signal or noise
sensor	R	station sensor (instrument) information
site	R	station location

▼ Architectural Design

TABLE 2: DATABASE TABLES USED BY SWIM SOFTWARE (CONTINUED)

Name	Mode	Description
sitechan	R	instrument orientation
stamag	R/W	station magnitude created by <i>EvLoc</i> , not <i>maxsurf</i> deleted if it exists by <i>MsOrid</i>
wfdisc	R	waveform information

FUNCTIONAL DESCRIPTION

The functions of the main components of the SWIM software are as follows:

- *LPcompile*
This small utility program converts regionalized dispersion curves to a binary format to improve *maxsurf* performance.
- *MsInterval*
This script collects event origins to be processed in a given time interval and then launches *MsOrid* for each origin.
- *MsOrid*
This script collects information for each event, starts *maxsurf* for events that have an m_b and a high probability of being within 200 km of the surface, and then starts *EvLoc* to calculate M_s .
- *maxsurf*
This program processes seismic data to identify surface waves and measure their properties.
- *MsConflict*
This script removes misassociated and spurious arrivals and starts *EvLoc* to compute M_s .

Figure 4 shows the processing flow of the SWIM software and *EvLoc*.

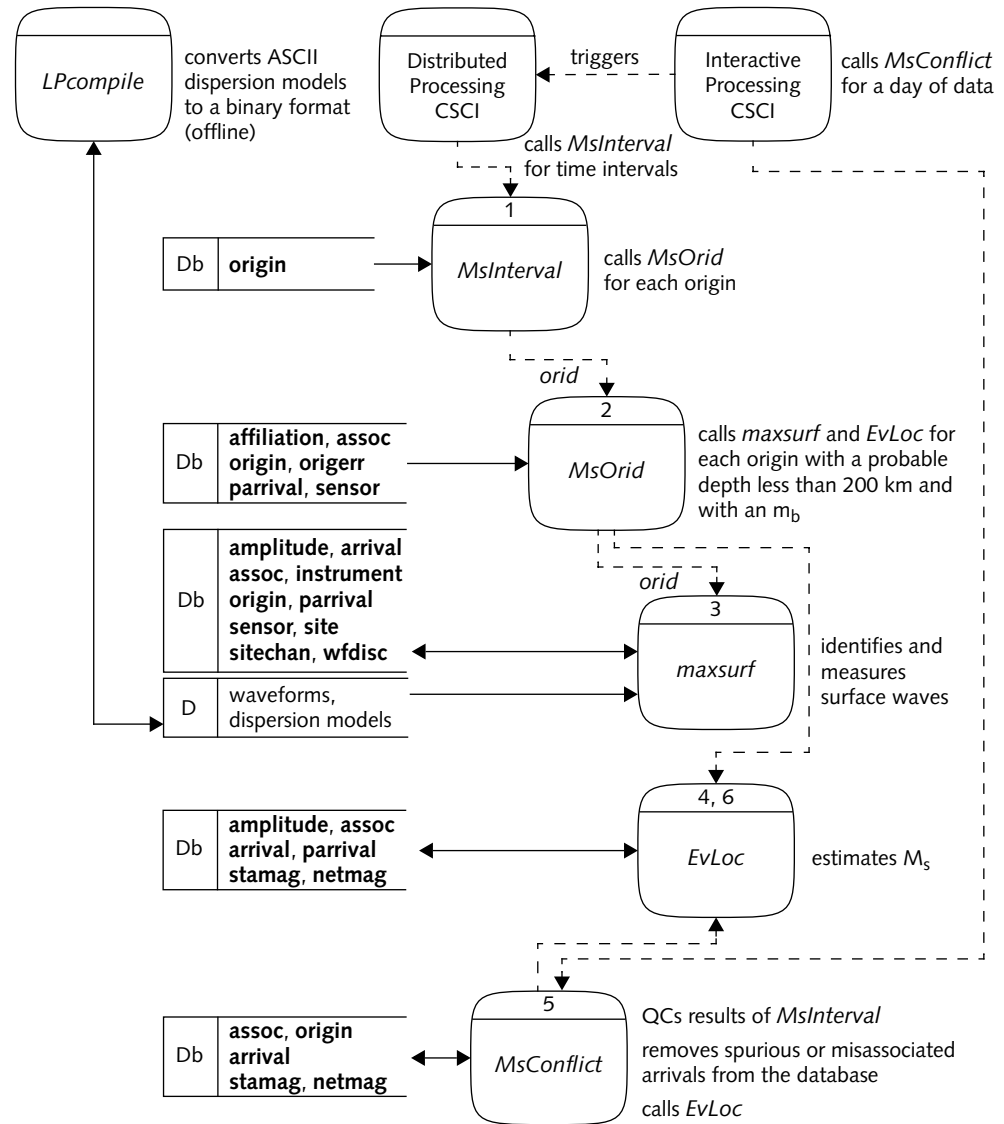


FIGURE 4. PROCESSING FLOW OF SWIM SOFTWARE

▼ Architectural Design

Creating Binary Files from Regionalized Dispersion Curves

Regionalized dispersion curves, used by the SWIM processing software, can require a significant amount of time to read when stored in ASCII format. *LPcompile* is used to convert regionalized dispersion curves to a binary format. The program must be run any time that the inputs to the program have been updated.

Determining Origins for Processing

When individual analysts finish reviewing an interval of data they start *MsInterval*, which begins the process of estimating surface wave magnitudes. *MsInterval* queries the **origin** database table to find all origins within the time interval reviewed by the analyst. For each origin, *MsInterval* calls *MsOrid*. *MsOrid* gathers information from the database and determines if the origin has an m_b and is shallow enough that surface waves might be detected (the depth of the event minus the uncertainty in the depth must be less than 200 km). If these conditions are met, then *MsOrid* provides the following information to *maxsurf*: the name of the parameter file, the database login account, the origin identifier (ID) for the event to be processed, the list of channels to be processed, and the list of stations to be processed.

maxsurf is run twice for each event, once for long-period data and once for broadband data. If a station has both long-period and broadband channels, and if *maxsurf* fails to find a surface wave in the long-period data, it will attempt to find a surface wave in the broadband data.

Identifying and Measuring Surface Waves

maxsurf automatically identifies and measures surface waves. Using the event location, *maxsurf* predicts the surface wave arrival times for each station and looks for surface waves within a time window corresponding to a group velocity window of 5 km/s to 2 km/s. Also, given a regionalized group velocity dispersion model, *maxsurf* predicts the arrival time of each frequency. The group arrival times are measured for each frequency, and then *maxsurf* checks for consistency with the predicted arrival times. This is the primary means of surface wave identification

used by *maxsurf*. Whether or not a surface wave is identified, *maxsurf* digitally replaces the instrument with a standard KS36000 long-period instrument and then searches for the largest amplitude with periods between 18–22 seconds. This amplitude is stored in the database identified as signal if a surface wave is found and as a noise measurement and upper bound on the surface wave size otherwise. If no amplitude within this period range and time window can be measured, then no information is written to the database.

Calculating Magnitudes

The *maxsurf* program calculates the surface wave magnitude, M_s , and uses it for an internal quality check, but *maxsurf* does not write the magnitude to the database. The magnitudes stored in the database are calculated by *EvLoc*, which calculates all station magnitudes associated with an event at the same time as the network magnitude and assigns these magnitudes the same *magid* key value. *EvLoc* is called by *MsOrid* after *maxsurf* has finished processing and by *MsConflict* after clean-up and quality checks are made.

Both *maxsurf* and *EvLoc* use the formula of Rezapour and Pearce [Rez98] to calculate magnitude:

$$M_s = \log(\text{amplitude}/\text{period}) + Q(\text{delta})$$

where Q is the distance correction.

Identifying and Removing Spurious and Misassociated Arrivals

MsConflict removes spurious or misassociated arrivals from the database and re-runs *EvLoc*. *MsConflict* tries to identify and remove two types of errors: 1) arrivals that are associated with more than one event, and 2) arrivals that are spurious and by chance pass the dispersion test. *MsConflict* runs four database queries, described as follows:

▼ Architectural Design

1. The first query looks for all events with four or more arrivals and makes a preliminary assessment that these arrivals are valid. It marks the arrival by changing the quality code to "g".
2. The second query checks to see if any of the arrivals marked with "g" are associated with more than one event. If found, these are marked with an "h". In general, such arrivals cannot be definitively associated with any event and are removed and treated as noise measurements.
3. All arrivals are checked to see if they are associated with more than one event. If one arrival is marked with "g" and others are not, then the arrivals not marked with "g" are assumed to be misassociated and are marked with "m." Arrivals associated with multiple events not marked with "g" cannot be definitively associated with any event and are removed and treated as noise measurements.
4. Arrivals at greater than 60 degrees from events that have no arrivals at distances less than 60 degrees are assumed to be spurious and removed. These arrivals are removed because the time windows at large distances can become quite long leading to a greater chance of a spurious arrival, and because in most cases if there are legitimate arrivals at large distances there are also arrivals at short distances. These arrivals are marked with "s".

After these queries are performed, all arrivals marked with "m", "h", or "s" are removed from the **arrival** and **assoc** tables. The remaining arrivals are all from events with four or more arrivals, or with arrivals recorded at less than 60 degrees, and all arrivals are uniquely associated with a single event.

INTERFACE DESIGN

This section describes the SWIM software interfaces with other IDC systems and with operators.

Interface with Other IDC Systems

The SWIM software's interface with other IDC systems is through the database, through the DACS, and through startup calls in the Interactive Processing CSCI.

When analysts complete reviews of their assigned time intervals, an interval is inserted into the **interval** database table. The DACS manages the processing using the **interval** table entries. *tuxshells* call *MsInterval*, which begins the surface wave processing. *MsConflict* is not managed by the DACS, but is started manually by the senior analyst just prior to preparing the REB.

The SWIM software extracts information from the database, and writes results to the database. The main SWIM software processing program, *maxsurf*, uses files stored on the system that describe instrument responses and regionalized dispersion curves and writes results for use by other IDC systems in the **arrival**, **assoc**, **par-rival**, and **amplitude** tables.

Interface with Operators

The SWIM software is designed to be an automatic processing module and therefore does not have an operator interface. The only normal interaction with operators is review of diagnostic output and occasional changes to the input parameter files.

Chapter 3: Detailed Design

This chapter describes the detailed design of the SWIM software and includes the following topics:

- Data Flow Model
- Processing Units
- Database Description

Chapter 3: Detailed Design

DATA FLOW MODEL

The SWIM software prepares static regional dispersion models for later processing, manages the processing, and processes the data.

The data flow model for *LPcompile*, the program that prepares regional dispersion models for later processing, is simple. ASCII files containing regional dispersion models and a grid file are read into the program and are written to a file in a binary format.

MsInterval and *MsOrid* manage the processing of surface wave data for specific time intervals of data (the DACS coordinates the time intervals). Given a time interval, these scripts extract origins from the database that are in the time interval and loop over these origins to determine which have an m_b and depth minus depth error less than 200 km. Origin and station information are extracted from the database for these origins and are passed to the main processing program, *maxsurf*.

The data flow of *maxsurf* is described by the following steps:

1. Read the command line input and the parameter file. This information defines the origin ID of the event to be processed, the stations in the network with long-period or broadband data to be processed, and various parameters needed for data processing. Arrays are described by their array name rather than individual station names.
2. Connect to the ORACLE database.
3. Extract event and network information from the database.
4. Loop over each station or array identifier. Each of the following steps occurs for each station.

5. Determine whether the data are from a 3-C station or an array by the number of stations associated with the identifier. Check the **affiliation** table to see if more than one *sta* entry is associated with the *net* entry for stations with an "lz" or "bz" channel. If so, then the station is an array.
6. Calculate the event-to-station distance. If the station is outside the limits defined in the parameter file, skip the station and continue processing with the next station. Current processing excludes stations greater than 100 degrees from the event.
7. Apply a minimum threshold test by predicting the network m_b above which surface waves can be reasonably expected at a distance of Δ degrees:

$$m_b \geq MAGBmin + 1.66 \log(\Delta / MAGBdelta)$$

The default values of *MAGBmin* and *MAGBdelta* are 2.5 and 20.0, respectively (the threshold is typically set to m_b 2.5 at 20 degrees for 3-C stations based on past experience with surface wave detection thresholds and should be reevaluated from time to time). Subtract 0.2 magnitude units from *MAGBmin* for array stations. If the m_b value is lower than the threshold, skip the station and continue processing with the next station.

8. Read waveform segments for all channels at each station and for all stations in an array.
9. Beam array data using vertical components for all long-period or broadband stations using the known azimuth to the event. The beam is typically formed using a constant phase velocity of 3.5 km/s; however, *maxsurf* also has the capability to use a regionalized phase velocity model to form the beam.
10. Remove the instrument response and replace it with a common long-period instrument. The new instrument response is an input quantity, but the operational system always uses a standard KS36000 long-period instrument.

▼ Detailed Design

11. Apply a set of narrow-band filters to the waveform: the vertical component at 3-C stations or the beam at arrays. This generates a set of surface wave arrival times at a set of frequencies, which are compared with a predicted dispersion curve. The frequencies are defined by input and are typically in the range of 0.02–0.06 Hz.
12. Calculate the predicted group velocity dispersion curve from a regionalized dispersion model and the source and receiver locations.
13. Apply the following test for surface wave existence: check the observed arrival time at each frequency to see if it is in a specified time window near the predicted arrival time. If a certain fraction of the measurements are within this window, then a surface wave is determined to exist in the waveform segment. This fraction is set by input and is typically 70 percent.
14. Estimate the station-to-event azimuth. For 3-C data use polarization analysis. For arrays form a beam over all azimuths and select the azimuth with the highest F-statistic.
15. Optionally, apply another existence test based on comparison of the estimated and known azimuths. This test is no longer used as it was found to be unreliable and to reject too many valid arrivals.
16. Determine the surface wave arrival time and amplitude by searching through all peaks in the waveform and finding the peak with the largest instrument corrected amplitude between 18 and 22 seconds. Measure the amplitude whether or not a surface wave is determined to exist (the amplitude is used as a noise measurement when calculating maximum likelihood magnitudes).
17. Calculate the surface wave magnitude and print it to standard output. Use the magnitude for two reasonableness tests; first, reject any M_s below a minimum threshold (typically M_s 1.0), and second, reject any M_s so large that it is inconsistent with m_b based on historical data. For example, reject an M_s of 5.5 or larger for an event with m_b 4.0. This condition

can occur when an arrival from a large earthquake is incorrectly associated with an aftershock closely spaced in time. M_s is calculated using the Rezapour and Pearce [Rez98] equation.

18. Write the results to the database. If a surface wave arrival is found, create entries in four database tables: write measured amplitudes to the **arrival** table, write the link between the arrival and the origin to the **assoc** table, write information about the waveform segment to the **parrival** table, and write the measured amplitude to the **amplitude** table. If no surface wave arrival is found, then write entries only to the **parrival** and **amplitude** tables.
19. After all stations are processed, close the connection to the database.

When all of the time intervals for a day of data have been processed and the results reviewed by the lead analyst, *MsConflict* is used to remove arrivals associated with more than one event and other spurious arrivals in the database. *MsConflict* then calls *EvLoc* to recompute the surface wave magnitudes of each event using the quality checked arrivals.

PROCESSING UNITS

The SWIM software consists of the following processing units:

- *LPcompile*
- *MsInterval*
- *MsOrid*
- *maxsurf*
- *MsConflict*
- *libLP*
- *liblpararray*
- *liblppar*
- *libmag*
- *libnbf*

▼ Detailed Design

- *libpathcor*
- *libpmotion*
- *libresp*
- *libs3obj*
- *libutilc*
- *libutilf*

The following paragraphs describe the design of these units, including any constraints or unusual features in the design.

LPcompile

LPcompile converts regionalized dispersion curves from ASCII format to a binary format and writes the results to a file. The conversion to binary format is necessary because ASCII format regionalized dispersion curves require too much time to read.

Input/Processing/Output

LPcompile is executed by the following command:

```
LPcompile LPdir gridfile velfile [compfile]
```

LPdir is the directory containing the ASCII format dispersion curves. *gridfile* is the dispersion file that describes the grid and associates a particular dispersion model with a particular location on the grid. *velfile* is the dispersion file that contains all of the frequency dependent dispersion curves. *compfile* is the name of the output binary file that contains the information in both *gridfile* and *velfile*. If *compfile* is omitted, then the output binary file is named *velfile.o*, where *velfile* is the input dispersion filename.

The *gridfile* contains the number of latitude and longitude samples and the spacing between geographic points at the top. This is followed by the indexes of the *velfile* dispersion model assigned to each grid point. The indexes are grouped by co-latitude range and are listed in increasing longitude range order. An example of a *gridfile* follows:

```
# #Lats  #Lons  Spacing (deg.) for grid indices
      36      72      5.0
```

```
# Co-Latitude Range:  0.00 to 5.00 deg.
```

```
150 150 150 150 150 150 150 150 150 150 150 150 150 150 150
150 150 150 150 150 151 151 151 151 151 151 151 150 150 150
150 150 151 151 151 151 151 151 151 151 151 129 129 129 129
129 151 151 151 151 151 151 151 151 151 151 151 151 151 151
151 151 151 151 151 151 151 151 151 151 151 151
```

```
# Co-Latitude Range:  5.00 to 10.00 deg.
```

```
150 150 151 151 151 151 151 151 113 113 113 113 113 113 151
151 151 151 151 151 151 151 150 150 150 151 151 151 151 151
151 151 139 139 139 151 151 151 151 151 151 151 151 151 151
139 139 139 139 139 139 139 139 139 124 43 43 43 43 43 44
44 44 44 44 45 44 44 43 23 151 151
```

```
# Co-Latitude Range: 10.00 to 15.00 deg.
```

```
2 21 118 118 23 24 24 25 25 26 26 25 25 25 25 25 24 23
118 118 118 23 112 113 113 112 23 23 23 23 112 112 113 139
139 151 151 23 24 139 139 139 139 139 139 139 115 122 122
122 122 122 124 43 43 118 24 43 44 47 47 48 48 47 46 45 44
25 26 24 3
```

... (one group for each co-latitude range up to 85.0 to 90.0 degrees)

▼ Detailed Design

The *velfile* first lists the number of model index samples and the number of period samples. This is followed by the periods (in seconds) that the period samples represent (for example, if there are 100 period samples, then 100 periods are listed). The remainder of the file consists of the models (one for each index) with the phase velocity at each of the period samples. An example of a *velfile* follows:

```
# LR parameter table discretized as index/period
154  # number of index samples
100  # number of period samples
5.000 5.051 5.102 5.155 5.208 5.263 5.319 5.376
5.435 5.495 5.556 5.618 5.682 5.747 5.814 5.882
5.952 6.024 6.098 6.173 6.250 6.329 6.410 6.494
6.579 6.667 6.757 6.849 6.944 7.042 7.143 7.246
7.353 7.463 7.576 7.692 7.812 7.937 8.065 8.197
8.333 8.475 8.621 8.772 8.929 9.091 9.259 9.434
9.615 9.804 10.000 10.204 10.417 10.638 10.870
11.111 11.364 11.628 11.905 12.195 12.500 12.821
13.158 13.514 13.889 14.286 14.706 15.152 15.625
16.129 16.667 17.241 17.857 18.519 19.231 20.000
20.833 21.739 22.727 23.810 25.000 26.316 27.778
29.412 31.250 33.333 35.714 38.462 41.667 45.455
50.000 55.556 62.500 71.429 83.333 100.000 125.000
166.667 250.000 500.000
# A0 normal oceanic, 0.15 km seds.
1.285697
1.282573
1.279342
1.276001
1.272544
1.268967
1.265263
1.261428
```

```
1.257454
1.253336
... (one sample for each period)

# A1      normal oceanic 0.5 km seds.
1.202370
1.200918
1.199333
1.197610
1.195747
1.193738
1.191579
1.189264
1.186789
1.184148
... (one sample for each period, one group for each model index)
```

Control

LPcompile is executed from the command line. It terminates when the binary file has been created.

Interfaces

LPcompile reads the two ASCII files and writes a binary file to the file system. The program must be run before the regional dispersion models can be used in *maxsurf*, but it has no direct interface to *maxsurf* itself. *LPcompile* is a stand-alone program that is run only when the software is initially installed and when the dispersion models are changed.

Error States

LPcompile fails if either of the input files do not exist or if these files have errors that cause read failures.

MsInterval

MsInterval gets event origins in a given time interval and calls *MsOrid* for each origin within the time interval.

Input/Processing/Output

MsInterval uses the *getpar.pl* Perl library to provide a standard par interface. *MsInterval* may be executed from the command line as follows:

```
MsInterval par=parfile
```

From *tuxshell*, *MsInterval* is called with the following command:

```
MsInterval par=Ms.par start-time=epochtime end-time=epochtime
```

For simplicity, most of the parameters are common to the *MsOrid*, *MsConflict*, and *MsInterval* scripts. Therefore, only one par file must be maintained. Because the par file is shared by three separate scripts, no defaults are supported to ensure that defaults do not differ between scripts. The parameters in the *MsOrid*, *MsConflict*, and *MsInterval* par file follow:

start-time

Epoch start time of data interval.

end-time

Epoch end time of data interval.

account

Database account.

out_arrival_table

Output **arrival** table.

out_assoc_table

Output **assoc** table.

out_netmag_table

Output **netmag** table.

out_origerr_table

Output **origerr** table.

out_origin_table

Output **origin** table.

out_parrival_table

Output **parrival** table.

out_stamag_table

Output **stamag** table.

out_amplitude_table

Output **amplitude** table.

magtype

Magnitude type written to the **netmag** table (single quoted).

phaselist

Valid phase types in the **assoc**, **parrival**, and **stamag** tables (single quoted, comma separated).

amptype

Valid amplitude types in the **amplitude** table (single quoted, comma separated).

excludelist

Stations excluded from processing (single quoted, comma separated).

bzonly_list

Stations for which the bz channel is used (single quoted, comma separated).

overwrite

Force recompute even if the *orid* has already been processed (0=no, 1=yes).

▼ Detailed Design

logprivate

Log database connection information (passwords) (0=no, 1=yes).
Extreme care should be used when using this option.

logargs

Write key arguments to the log, *start-time*, *end-time*, *account*, script bin directory, par file directory, and *out_origin_table* name (0=no, 1=yes).

logquery

Write SQL commands to the log file prior to execution (0=no, 1=yes).

logentry

Log information regarding major routines such as *EvLoc* and *maxsurf* (0=no, 1=yes).

logsubs

Log major script milestones (0=no, 1=yes).

logmaxsurf

Write all *maxsurf* generated messages to the log file (0=no, 1=yes).

logEvLoc

Write all *EvLoc* generated messages to the log file (0=no, 1=yes).

An example par file follows:

```
#          %G%          %W%
# PAR file for MsInterval and MsOrid
#
# Database information
account=LATEDB
out_arrival_table=OUT_ARRIVAL
out_assoc_table=OUT_ASSOC
out_netmag_table=OUT_NETMAG
out_origerr_table=OUT_ORIGERR
out_origin_table=OUT_ORIGIN
out_parrival_table=OUT_PARRIVAL
out_stamag_table=OUT_STAMAG
out_amplitude_table=OUT_AMPLITUDE
```

```

#
# Valid Phase/Magnitude types
magtype='ms_ave'
phaselist='LR'
amptype='ANL/2','LR'
#
# Stations excluded from Processing
excludelist='ASAR','TXAR'
#
# Stations for which bz channel is used
bzonly_list='HIA','BJT'
#
#
# Do not force recompute if orid already processed
overwrite=0
#
# Logging verbosity
logprivate=0 #logprivate=1 will log DB connect info.
logargs=1    #Log key command arguments
logquery=0   #Log queries
logentry=0   #Log entry and exit to Ms scripts
logsubs=0    #Log major routines within Ms scripts
logmaxsurf=1 #Log output of maxsurf
logEvLoc=1   #Log output of EvLoc
#

```

Control

MsInterval is called directly from the *tuxshell* controlling surface wave processing.

Interfaces

MsInterval interacts with the ORACLE database through SQL statements (select directly, update and delete indirectly through *MsOrid*).

▼ Detailed Design

Error States

MsInterval detects any unusual exit status from *MsOrid*, writes an appropriate message to the log file, and exits with a non-zero status. A non-zero exit of *MsInterval* will signal the *tuxshell* to take appropriate action, either re-queuing the interval or marking the interval as `ms-failed` (through `interval.state`).

The logging support offered by the three scripts for surface wave processing offers nearly complete control over the messages that will be written to the log file. Logging parameters may be either off (0) or on (1).

MsOrid

MsOrid controls the processing of surface wave data for a given origin. *MsInterval* (see “*MsInterval*” on page 32) provides *MsOrid* with event origins to be processed. *MsOrid* executes two programs: *maxsurf*, which identifies and measures surface waves, and *EvLoc*, which calculates surface wave magnitudes from the amplitude data generated by *maxsurf*.

Input/Processing/Output

MsOrid uses the *getpar.pl* Perl library to provide a standard par interface.

For simplicity, most of the pars are common to the *MsOrid*, *MsConflict*, and *MsInterval* scripts, and therefore, only one par file must be maintained (see “*MsInterval*” on page 32). Because the par file is shared by three separate scripts, no defaults are supported to ensure that defaults do not differ between scripts.

The first *MsOrid* processing step is to remove old data in the **parrival**, **amplitude**, **arrival**, **assoc**, **stamag**, and **netmag** tables. Depth and m_b magnitudes are checked; if the epicenter is too deep or no m_b magnitude is available then no surface wave processing is carried out. *MsOrid* loops through long-period and broadband station groups separately and applies *maxsurf* to each group. Stations may be either 3-C or array.

After verifying that the surface wave measurements have been made the surface wave magnitudes are calculated by initiating *EvLoc*.

The final processing step is to update *origin.nass* (the number of associated arrivals) in accordance with the results of the processing.

Control

MsOrid is called directly from *MsInterval*.

Interfaces

MsOrid interacts with the ORACLE database through SQL statements (select, update, and delete).

Error States

MsInterval detects any unusual exit status from *MsOrid*, writes an appropriate message to the log file, and exits with a non-zero status.

The logging support offered by the three scripts for surface wave processing (*MsInterval*, *MsOrid*, and *MsConflict*) offers nearly complete control over the messages that will be written to a log file (see the parameters beginning with “logprivate” on page 34).

MsOrid uses the *ora_select.pl* Perl library to trap ORACLE errors allowing *MsOrid* to exit with a non-zero status if any query fails. In addition, the execution of *maxsurf* and *EvLoc*, as children of *MsOrid*, is monitored, and a non-zero exit by either application will cause *MsOrid* to exit with a non-zero status.

maxsurf

maxsurf is the main SWIM software program that identifies and measures surface waves. This section describes the structure of the code and how it interfaces with other libraries. Most of the libraries used by *maxsurf* are designed as objects that

▼ Detailed Design

maintain their own internal data and are accessed through a set of object methods. Although the modules are written in C, they are constructed in much the same manner as C++ objects. This design feature improves the modularity and maintainability of the code. An overview of each object is provided in the library descriptions.

Input/Processing/Output

maxsurf can be run using the following command:

```
maxsurf par=parfile
```

parfile is the standard format parameter file containing all *maxsurf* input parameters. When *maxsurf* is started by *MsOrid*, the *parfile* contains only the information that is static for each event. The origin ID, station identifiers for long-period or broadband channels, and database account are passed as command line arguments as follows:

```
maxsurf par=parfile net_list=nlist orid=orid chan_list=clist \  
in_db=acct
```

The following parameters are applicable to *maxsurf*:

DECfilters

List of decimation filters to be applied to the data. Decimation filters are chosen from SAC2, SAC3, SAC4, SAC5, SAC6, and SCAT4, where SAC[2–6] are decimation filters derived from the Seismic Analysis Code (SAC) and represent decimation by factors of 2–6, respectively, and SCAT4 is a shorter, faster filter for decimation by 4. Default: (see *DECsamprate*).

DECsamprate

Desired sampling rate (samples/second). *maxsurf* uses the decimation filters (see *DECfilters*) to come close to this sampling rate.

Default: 1.

LPAarray

Positive values indicate that the input data are to be treated as an array. All stations used for array processing must have the same reference station listed in the **site** table. By default, the station name used for array results is the reference station. The station name may be changed by specifying *out_station_name*. Default: 0.

LPAazmax

Maximum azimuth value (degrees) used in the LP array beam search. Default: 360.0.

LPAazmin

Minimum azimuth value (degrees) used in the LP array beam search. Default: 0.0.

LPAaznum

Number of azimuth values used in the LP array beam search. Default: 73 (every 5 degrees).

LPAbeam3c

When positive, 3-C beaming will be used for further processing. Set to zero to beam vertical component only. Default: 0.

LPAbeamEllip

Ellipticity used to form the long-period beam for further processing. This should be the ellipticity corresponding to the earth structure near the array, not the path. A constant (frequency-independent) value should be adequate. Used only for 3-C beam forming. Default: 0.8. Beam ellipticity and phase velocity can also be specified as frequency-dependent arrays of type float using the input parameters *LPAbeamFreqArr*, *LPAbeamEllipArr*, and *LPAbeamVelArr*.

▼ Detailed Design

LPABeamEllipArr

Frequency-dependent ellipticity used to form the long-period beam for further processing. Each member of the list corresponds to the frequency set by *LPABeamFreqArr*. The ellipticity corresponds to the earth structure near the array, not the path. See *LPABeamVelArr*.

LPABeamFreqArr

Frequencies (Hz) used in frequency-dependent ellipticity and phase velocity specifications. See *LPABeamEllipArr* and *LPABeamVelArr*.

LPABeamRegional

When positive, a regionalized phase velocity model will be used for beamforming instead of a fixed phase velocity. See *LPABeamVel* and *LPABeamVelArr*. The phase velocity is specified in the same format as the regionalized dispersion curve in directory *PCORregdir*, with name *LP_pvel.LR* specified by the grid file *LP_grid.LR*. The two files may be combined in binary form as *LP_pvel.LR.o*. Default: 0.

LPABeamVel

Phase velocity (km/second) used to form the long-period beam used for further processing. This should be the phase velocity across the array, not along the path. A constant (frequency-independent) value should be adequate. Default: 3.5. Beam phase velocity and ellipticity can also be specified as frequency-dependent arrays of type float using the input parameters *LPABeamFreqArr*, *LPABeamEllipArr*, and *LPABeamVelArr*.

LPABeamVelArr

Frequency-dependent phase velocity (km/second) used to form the long-period beam for further processing. Each member of the list corresponds to the frequency set by *LPABeamFreqArr*. The phase velocity is across the array, not along the path. See *LPABeamEllipArr*.

LPAdelStations

When positive, any arrivals that exist in the database for individual stations in the array as well as arrivals for the reference stations will be deleted. Default: 1.

LPAelmax

Maximum ellipticity value used in the LP array beam search.

Default: 0.8.

LPAelmin

Minimum ellipticity value used in the LP array beam search.

Default: 0.8.

LPAelnum

Number of ellipticity values used in the LP array beam search. Default: 1.

LPAfreq

Set of frequencies (Hz) used to form the beam for the long-period array azimuth search. Default: 0.02, 0.025, 0.03, 0.035, 0.04, 0.05, 0.06. All values must be greater than zero.

LPAhorWeight

Sets the weighting of all horizontal component data used in array processing. If set to zero, only vertical component data will be used. If *LPAbeam3c* is set to zero while *LPAhorWeight* is positive then 3-C data will be used for the azimuth search, but only vertical component data will be used for forming the beam for additional processing.

Default: 1.0.

LPAverWeight

Sets the weighting of all vertical component data used in array processing. *LPAverWeight* and *LPAhorWeight* are scale independent. That is, for example, weights of 10.0 and 1.0, respectively, will give the same results as 1.0 and 0.1. Default: 1.0.

LPAtestAz

When positive, the long-period array test for measured azimuth close to the actual back azimuth will be applied. Default: 0. See *azdiffmax* and *azshift*.

▼ Detailed Design

LPAtestFstat

When positive, the long-period array test for minimum F-statistic will be applied. Default: 0. See *fstatmin*.

LPAvpmax

Maximum phase velocity (km/second) used in the LP array beam search. Default: 4.5.

LPAvpmin

Minimum phase velocity (km/second) used in the LP array beam search. Default: 2.5.

LPAvpnum

Number of phase velocity values used in the LP array beam search. Default: 21. The LP array beam data are searched on even intervals of slowness between $1/LPAvpmax$ and $1/LPAvpmin$.

NumFailAllowed

Set to non-zero to allow for a fixed number of test failures. Default: 0.

NBFdo

When positive, narrow-band filters will be applied to measure group velocity dispersion. Default: 1.

NBFfilterQ

Filter Q used for narrow-band filter analysis. Default: 15.0.

NBFfreq

Set of frequencies (Hz) used for narrow-band filtering. Default: 0.02, 0.025, 0.03, 0.035, 0.04, 0.05, 0.06. The default frequencies should always be used for *NBFtestNorth*. All values must be greater than zero.

NBFnoinst

When positive, the instrument group delay from narrow-band filtering is not corrected. This is a time-consuming operation and a small correction if the instrument phase is slowly varying. Default: 0.

NBFresultsdir

Directory to write predicted and observed group velocity dispersion curves for each event/station pair. No default. If not given, dispersion curves are written to standard output.

NBFtestDispersion

When positive, the group velocity arrival times are tested for consistency with predicted group velocities. The group velocity arrival times are determined by narrow-band filtering. The predicted arrival time may either be specified as input frequencies and velocities *fdisp* and *vdisp* or derived from a regionalized group velocity model (see *PCORregdir*). The expected arrival time interval is specified by defining a time window around the predicted group velocity time. This may be specified as a group velocity range, time range, or period-dependent range using the parameters *vdisplimit*, *tdisplimit*, and *pdisplimit*, respectively. The measured group arrivals are required to fall within the time interval:

$$t_{\min} = \frac{\text{distance}}{(v_{\text{disp}} + v_{\text{displimit}})} - (p_{\text{displimit}} \cdot \text{period}) - t_{\text{displimit}}$$

and

$$t_{\max} = \frac{\text{distance}}{(v_{\text{disp}} - v_{\text{displimit}})} + (p_{\text{displimit}} \cdot \text{period}) + t_{\text{displimit}}$$

The default values of these parameters are *vdisplimit*=0.2, *pdisplimit*=1.0, and *tdisplimit*=0.0. For example, if the expected group velocity of a 20 second arrival is 3.0 km/second and the distance from event to station is 6000 km, the default time window is from 1855 to 2163 seconds after origin time. A surface wave passes this test if the fraction of points determined at different frequencies satisfying this condition exceeds *NBFtestDispFrac*. See *NBFtestDispFrac*, *fdisp*, *vdisp*, *vdisplimit*, *pdisplimit*, and *tdisplimit*. Default: 0.

NBFtestDispFrac

Fraction of narrow-band filter group velocity estimates that must lie within the predicted arrival time window to pass the dispersion test. See *NBFtestDispersion*, *fdisp*, *vdisp*, and *vdisplimit*. Default: 0.7.

NBFtestNorth

When positive, the North and Woodgold [Nor94] test for surface wave dispersion is applied. This test looks for decreasing group velocity with increasing frequency and allows for one exception at the lowest or highest frequency and one exception in between. Default: 0.

PCORcompiled

When positive, regionalized dispersion curves will be read in binary rather than ASCII format. The file *LP_vel.LR.o* must exist in the directory *PCORregdir*. This can lead to a substantial savings in time for detailed dispersion models. The files can be compiled to binary form with the program *LPcompile*. Default: 0.

PCORregdir

Directory containing regionalized group velocity curves. The directory must contain two files named *LP_grid.LR* and *LP_vel.LR* specifying grid points and group velocities for Rayleigh waves. The two files may be combined in binary form as *LP_vel.LR.o*. Optional. No default.

POLdo

When positive, polarization filtering to estimate station to event azimuth is applied. Default: 1.

POLfmax

Ending frequency (Hz) for polarization filtering. Default: 0.05.

POLfmin

Starting frequency (Hz) for polarization filtering. Default: 0.02.

POLnoinst

When non-zero, the instrument response correction is not applied prior to polarization filtering. This is primarily an optimization feature, because instrument correction can be quite time consuming in some cases.

Default: 0.

POLtestAz

When positive, the polarization test for measured azimuth close to actual back azimuth is applied. Default: 0. See *azdiffmax*.

POLtestFstat

When positive, the test for F-statistic of the polarization test is applied.

Default: 0. See *fstatmin*.

POLtestEllip

When positive, the test for reasonable ellipticity in the polarization test is applied. Default: 0. See *elmin* and *elmax*.

POLtestAll

When positive, *POLtestAz*, *POLtestFstat*, and *POLtestEllip* are combined into a single test.

MAGdo

When positive, a surface wave arrival is found and a magnitude is calculated. Default: 1.

▼ Detailed Design

MAGBtest

Prunes out unreasonable surface wave amplitudes based on two tests against the m_b for the event. Can only be applied if m_b is defined in the **origin** table. The first test is a simple threshold test based on magnitude and distance, rejecting an arrival if

$$m_b < MAGBmin + 1.66 \log(\Delta/MAGBdelta)$$

where the default values of *MAGBmin* and *MAGBdelta* are 2.5 and 20.0, respectively. The second test removes arrivals where M_s is larger than all historical values of M_s for events of the same m_b . Arrivals are removed if M_s exceeds m_b+2 for $m_b < 5$, $m_b+1.5$ for $m_b < 4.5$, and m_b+1 for $m_b < 4$.

MAGBdelta

Distance (degrees) corresponding to *MAGBmin* if *MAGBtest* is applied.

Default: 20.0.

MAGBmin

Minimum magnitude (m_b) at *MAGBdelta* for which a surface wave arrival will be allowed. Default: 2.5. See *MAGBtest*.

MAGdfac

"D" in the equation for M_s , as defined by Rezapour and Pearce [Rez98]:

$$M_s = \log\left(\frac{A}{T}\right) + k \cdot \log(\Delta) + 0.5 \cdot \log(\sin(\Delta)) + \gamma \cdot \Delta \cdot \log(e) + D$$

where k is 1/3 or 1/2 depending on whether the arrival is an Airy phase or normally dispersed. The Rezapour and Pearce values are $k=1/3$, $\gamma=0.0105$, $D=2.370$. These values are set with the parameters *MAGkfac*, *MAGgamma*, and *MAGdfac*, respectively, which default to the Rezapour and Pearce values. Default: 2.370.

MAGgamma

" γ " in the equation for M_s , as defined by Rezapour and Pearce [Rez98].

$$M_s = \log\left(\frac{A}{T}\right) + k \cdot \log(\Delta) + 0.5 \cdot \log(\sin(\Delta)) + \gamma \cdot \Delta \cdot \log(e) + D$$

where k is 1/3 or 1/2 depending on whether the arrival is an Airy phase or normally dispersed. The Rezapour and Pearce values are $k=1/3$, $\gamma=0.0105$, $D=2.370$. These values are set with the parameters *MAGkfac*, *MAGgamma*, and *MAGdfac*, respectively, which default to the Rezapour and Pearce values. Default: 0.0105.

MAGkfac

" k " in the equation for M_s , as defined by Rezapour and Pearce [Rez98]:

$$M_s = \log\left(\frac{A}{T}\right) + k \cdot \log(\Delta) + 0.5 \cdot \log(\sin(\Delta)) + \gamma \cdot \Delta \cdot \log(e) + D$$

where k is 1/3 or 1/2 depending on whether the arrival is an Airy phase or normally dispersed. The Rezapour and Pearce values are $k=1/3$, $\gamma=0.0105$, $D=2.370$. These values are set with the parameters *MAGkfac*, *MAGgamma*, and *MAGkfac*, respectively, which default to the Rezapour and Pearce values. Default: 0.33333.

MAGtest

When positive, a test for reasonableness of magnitude is applied.

Default: 0.

MAGmax

Maximum allowed value (M_s) for magnitude test. Default: 10.0. See *MAGmin* and *MAGtest*.

MAGmin

Minimum allowed value (M_s) for magnitude test. Default: 2.0. See *MAGmax* and *MAGtest*.

MAGperMin

Minimum allowable period (seconds) for a magnitude measurement.

Default: 10.0.

▼ Detailed Design

MAGperMax

Maximum allowable period (seconds) for a magnitude measurement.

Default: 30.0.

MAGperPref

Preferred period (seconds) for a magnitude measurement.

Default: 20.0.

MAGperSlope

Specifies a period weighting function to be applied to determine the peak that corresponds to the desired arrival. The weighting function has the form:

$$W = \text{MAGperSlope} \cdot \text{ABS}(\text{period} - \text{MAGperPref})$$

The arrival is then found as follows: A magnitude is calculated for each peak in the wavetrain satisfying the arrival time test (see *vdisp*) and having a period between *MAGperMin* and *MAGperMax*. The weighting function *W* is then subtracted from each magnitude, and the largest weighted value is assumed to correspond to the surface wave arrival. Default: 0.2. The default value is weighted rather strongly toward 20 seconds to identify the best 20 second arrival. If *MAGperSlope* is set to zero, then there is no weighting, and the maximum value between *MAGperMin* and *MAGperMax* will be set as the surface wave arrival. See *vdisplimit*.

MAGtype

Surface wave magnitude type. *MAGtype* must be one of "MsGutenberg", "MsPrague", "MsVonSeggern", "MsMarshall-Basham", "MsTheoretical", or "MsRP", which correspond to the original Gutenberg magnitude, the Prague (IASPEI) variant of Gutenberg's magnitude, Von Seggern's M_s , which has an attenuation correction of 1.0 instead of 1.66, the Marshall and Basham [Mar72] M_s based on theoretical Rayleigh wave attenuation, and M_s as defined by Rezapour and Pearce [Rez98], which uses the theoretical formula with empirical values for the constants. *MsTheoretical* has the form:

$$M_s = \log\left(\frac{A}{T}\right) + k \cdot \log(\Delta) + 0.5 \cdot \log(\sin(\Delta)) + \gamma \cdot \Delta \cdot \log(e) + D$$

where k is 1/3 or 1/2 depending on whether the arrival is an Airy phase or normally dispersed. The Rezapour and Pearce values are $k=1/3$, $\gamma=0.0105$, $D=2.370$. For `MsTheoretical`, these values are set with the parameters `MAGkfac`, `MAGgamma`, and `MAGdfac`, respectively, which default to the Rezapour and Pearce values. The Marshall Basham magnitude corrects for path type and depth as well as distance. Default: "MsPrague". See `MAGpathtype`.

MAGpathtype

Path type used for the Marshall Basham magnitude. It must be one of "Continental", "Oceanic", "Mixed", or "Eurasia". This applies a path-dependent period correction to the magnitude. Not used for other magnitudes. Default: `Continental`.

Timer

When positive, the time that each part of the analysis takes is printed. Default: 0.

amptype

Amplitude type that will be placed in `amplitude.amptype`. Default: `ALR/2`.

arrival_auth

Author entry that will be placed in `arrival.auth`. Default: program name and version number.

arrival_name

Phase name that will be placed in `arrival.iphase` and `assoc.phase`. Default: `LR`.

azdiffmax

Maximum allowed difference (degrees) between the apparent azimuth and the actual backazimuth in the polarization and long-period array azimuth tests. Default: 30.0.

▼ Detailed Design

azshift

Expected azimuth shift (degrees) for a given path. That is, if an arrival consistently comes in 10 degrees off for a given path, *azshift* should be set to 10. Default: 0.0.

chan_list

List of channels of interest to be used in the **wfdisc** query. Required. Must be a 3-C list to use polarization filtering, for example:
'lz','ln','le'. (*libwfm*)

deltamin

Minimum distance (degrees) for which surface waves are processed.
Default: 0.0.

deltamax

Maximum distance (degrees) for which surface waves are processed.
Default: 180.0.

demean

When positive, the data are demeaned prior to processing. Default: 1.

detrend

When positive, the data are detrended prior to processing. Default: 1.

duration

Desired time length (seconds) from the *start_time*. A value is required because of data checks by the *libwfm* waveform routines. However, this parameter is superseded by the group velocity window parameters and is not used. (*libwfm*)

elmin

Minimum allowed value for the ellipticity test. Default: 0.4. See *elmax* and *POLtestEllip*.

elmax

Maximum allowed value for the ellipticity test. Default: 2.0. See *elmin* and *POLtestEllip*.

fdisp

Array of frequencies (Hz) corresponding to the group velocity dispersion in *vdisp*. Limited to 100 values. Default not used. All values must be greater than zero. Not needed if *PCORregdir* is given. See *vdisplimit*, *pdisplimit* and *tdisplimit*.

fill_value

Value put in the new waveform everywhere a gap occurs. Must be set to 0.0 to fill any small gaps in the data. (*libwfm*)

fractaper

Fraction of time series to taper prior to processing. Tapering is applied to both ends of the time series. Default: 0.025.

fstatmin

Minimum allowed F-statistic for polarization filtering and long-period array processing. Default: 1.0. See *POLtestFstat* and *LPAtestFstat*.

gvmax

Maximum group velocity (km/second). Defines the start of the time window to be processed. Default: 5.0.

gvmin

Minimum group velocity (km/second). Defines the end of the time window to be processed. Default: 2.0.

in_affiliation

Table name for obtaining stations affiliated with a given network. Default: *affiliation*. Optional. See *net*. (*libwfm*)

in_db

Input database name/password that contains the input **wfdisc** table. Either *in_db* or *in_file* is required but not both. (*libwfm*)

in_file

Directory/filename that contains the input wfdisc records. Either *in_file* or *in_db* is required but not both. (*libwfm*)

▼ Detailed Design

in_table

Database table name where input **wfdisc** tuples may be found. Default: **wfdisc**. Optional. (*libwfm*)

instfmax

Upper frequency (Hz) limit for instrument response corrections. Data are set to zero at higher frequencies. Instrument response corrections are performed for array analysis, narrow-band filtering, polarization filtering, and magnitude measurement. This option can save considerable CPU time with undecimated broadband data. Default: 0.5.

instrument_table

Database table containing instrument information.
Default: **instrument**.

join

Set to 1 to connect adjacent waveform segments. Time gaps are filled with values defined by *fill_value*. (*libwfm*)

keep_old

When positive, old database entries are not be deleted prior to adding new ones. Default: 0.

net

Name of the network that is used to obtain the stations of interest from the **affiliation** table. If only a single station in **affiliation** is linked to *net*, then *net* becomes the output station name. If multiple stations are associated with the *net*, then they are treated as an array. For long-period arrays, *net* or *sta_list* must define the elements of the array to be processed. Either *net* or *sta_list* is required but not both. See *net_list*. (*libwfm*)

net_list

List of station or array names that are treated as described in *net*. *net_list* may contain a mixture of 3-C stations and arrays. Because of the *libwfm* requirement that either *sta_list* or *net* be set, if using *net_list*, also set *net=DUMMY* and do not use *sta_list*. *net* is not used if *net_list* is given.

newinstcalib

Amplitude (nanometers/digital count) of the new instrument at the calibration period. See *newinstcalper*.

newinstcalper

Calibration period (seconds) of the new instrument. If not given, the response is not scaled. See *newinstcalib*.

newinstdir

Directory of an instrument file to replace the recording instrument for magnitude measurement. Optional. See *newinstfile*.

newinstwaterlevel

Lowest meaningful amplitude level (digital counts/nanometer) of old instrument, added to old response before dividing. Default: $1 \cdot e-4$.

newinstfile

Instrument filename to replace the recording instrument for magnitude calculations. This filename must be used for broadband data for magnitude to be meaningful. Optional. See *newinstdir*.

noise_amptype

Amplitude type used as **amplitude.amptype** for a noise arrival. Default: **ANL/2**. See *noise_phase* and *write_noise*.

noise_phase

Noise phase name used in **parrival.phase** entries for a noise arrival. Default: **LR**. See *noise_amptype* and *write_noise*.

orid

Origin ID of the event to be analyzed. No default. Required.

▼ Detailed Design

origin_db

Database containing origin and station information. Use this parameter only if the data are specified through a wfdisc file instead of a table so that *in_db* is not specified. No default.

origin_table

Database table containing origin information. Default: *origin*. See *orid*.

out_amplitude_table

Database **amplitude** table to which results are written. If the table is not specified, amplitudes are not written to the database. If *out_amplitude_table* is specified, **parrival** and **arrival** tables must also be specified to avoid missing database links. Amplitudes are written for both noise and signal measurements. No default. Optional.

out_arrival_table

Database **arrival** table to which results are written. If the table is not specified, arrivals are not written to the database. Arrivals are written only for signal measurements, not noise measurements. No default. Optional.

out_assoc_table

Database **assoc** table to which results are written. If the table is not specified, assoc rows are not written to the database. Assoc rows are written only for signal measurements, not noise measurements. No default. Optional.

out_parrival_table

Database **parrival** table to which results are written. If the table is not specified, parrival rows are not written to the database. Parrival rows are written for both noise and signal measurements. No default. Optional.

out_station_name

Alternate station name to be used for output results. This option can be used to replace a station name such as ARE0 with an array name such as ARCES, or to replace an array reference station name such as CM31 with an array name such as CMAR. The recommended procedure is to specify *net* or *net_list* instead of *out_station_name*. No default. Optional.

override_accept

When positive, the surface wave arrival is accepted and written to the database even if it fails the surface wave acceptance tests. This option is provided for those cases where an analyst can identify a surface wave that for one reason or another fails the tests applied by *maxsurf*.

override_reject

When positive, the surface wave arrival is rejected and not written to the database (while removing it if it has been added previously) even if it passes the surface wave acceptance tests. This option is provided for those cases where an analyst believes that the surface wave tests were passed fortuitously and a surface wave from the event being analyzed does not actually exist in the time window.

pdisplimit

Acceptable range for the number of periods that arrival times may be outside of the predicted dispersion curve. Arrivals are limited to the time window specified in *NBFtestDispersion*. Default: 1.0. See *vdisplimit*, *tdisplimit*, and *vdisp*.

read_by_station

If a long station list or network is given as input, *libwfm* will read all of the data into memory. Particularly with broadband data, this may cause the program to grow to a very large size. Setting this flag to 1 forces the program to read and process one station at a time. With this option a new query must be performed for each station. Optional. Default: 1.

▼ Detailed Design

results_dir

Directory to which a machine-readable summary of results will be written (if specified). Optional. No default.

sensor_table

Database **sensor** table containing instrument information.
Default: `sensor`.

site_table

Database **site** table containing station information. Default: `site`.

sitechan_table

Database **sitechan** table containing instrument orientation information.
Default: `sitechan`.

sta_list

List of stations of interest to be used in the **wfdisc** query. Either *sta_list* or *net* is required but not both. For long-period arrays, *sta_list* must correspond to the elements of the array to be processed. (*libwfm*)

start_time

Beginning epoch time of the waveform window. As with duration, a value is required, but not used. (*libwfm*)

tdisplimit

Acceptable range of arrival times (seconds) outside of the predicted dispersion curve. Arrivals are limited to the time window specified in *NBFtestDispersion*. Default: 0.0. See *vdisplimit* and *pdisplimit*.

vdisp

Array of group velocities (km/second) giving the predicted dispersion. Must be used together with *fdisp*. Limited to 100 values. Default not used. All values must be greater than zero. Not needed if *PCORregdir* is given. See *vdisplimit*, *pdisplimit*, and *tdisplimit*.

vdisplimit

Acceptable group velocity (km/second) range for which arrival times may be outside of the predicted dispersion curve. Arrivals are limited to the time window specified in *NBFtestDispersion*. Default: 0.2. See *pdisplimit* and *tdisplimit*.

vmodel

Velocity model name to place in **assoc** and **pararrival** tables. Default: "-".

wfd_to_wfm

Boolean flag that if positive indicates that the waveforms are to be stored in memory. This parameter must be true and cannot be omitted. (*libwfm*)

write_noise

When positive, a noise arrival is written to the database if a surface wave is not identified but the measured amplitude is greater than zero. Default: 0. See *noise_phase* and *noise_amptype*.

An example of a *maxsurf* par file follows:

```
# @(#)maxsurf.par      1.1 04/21/98
#
par=$(IMSPAR)
par=$(AUTOMATIC)
in_db=$(LATEDB)

# print out time spent in various modules
Timer=1

LPAbeamEllip=0.8
LPAbeamVel=3.5
LPAbeam3c=0
LPAhorWeight=1.0
LPAverWeight=1.0

#MAGB parameters
```

▼ Detailed Design

```
MAGBtest=1
MAGBdelta=20.0
MAGBmin=2.5

#MAG parameters
MAGmax=10.0
MAGmin=2.0
MAGperSlope=0.
MAGperMin=18.
MAGperMax=22.
MAGtest=1

#NBF parameters
NBFfilterQ=15
NBFfreq=0.02,0.025,0.03,0.035,0.04,0.045,0.05,0.06
NBFtestDispersion=1

NumFailAllowed=0

# compiled dispersion curves
PCORregdir=$(STATICDIR)/LPdisp/LP
PCORcompiled=1

POLtestAll=0
azdiffmax=45.0
arrival_name=LR

deltamax=100.0
deltamin=0.0
demean=1
detrend=1
duration=100000.0

fill_value=0.0
gvmax=5.0
gvmin=2.0
```

```
in_affiliation=affiliation
in_table=wfdisc
instrument_table=instrument
join=1

# net or sta_list must be given even though not used
net=DUMMY
newinstcalib=1.0
newinstcalper=20.
newinstdir=$(STATION-DIR)/rsp
newinstfile=KS36000.lp

origin_table=origin
out_arrival_table=arrival
out_assoc_table=assoc
out_amplitude_table=amplitude
out_parrival_table=parrival
sensor_table=sensor
site_table=site
sitechan_table=sitechan
start_time=0.0

pdisplimit=1.0
tdisplimit=0.0
vdisplimit=0.2

wfd_to_wfm=1

# write noise to database
write_noise=1

# set amptypes
amptype=ALR/2
noise_amptype=ANL/2
```

▼ Detailed Design

```
# keep old LR's. (MsOrid deletes them, so it is safe to
# "keep" them here.
keep_old=1
```

Control

maxsurf is started from the command line. In the operational system, the command is given from *MsOrid*, which provides an origin ID, station and channel IDs, and the database account to *maxsurf*. *maxsurf* terminates when all stations for the event are processed.

Interfaces

The interfaces to *maxsurf* are the command line and the database.

Error States

maxsurf exits with an error message if it cannot read the parameter file. If this occurs, it is probably because an incorrect filename was passed to *maxsurf*. If the database connection cannot be established, *maxsurf* does not immediately exit, but instead tries to gather all of the information it requires from the parameter file. If all of the required information is not available to process a given waveform segment, or if the data cannot be found or read, then *maxsurf* writes diagnostic output and continues with the next station.

Incorrect results may be obtained from *maxsurf* if the input information and data are incorrect or are not as assumed. For example, *maxsurf* corrects for off-angle orientations using information in the **sitechan** table. This will only work if the **sitechan** table is accurate. It is possible to correct implicitly for orientation shifts (but not polarity reversals) by using the *azshift* parameter.

The 3-C data are assumed to be all of the same type (for example, long-period or broadband). Either one (vertical) or three (vertical, north, east) channels must be specified.

Sampling rates must be the same for all waveforms processed at the same time. If sampling rates differ, they can be made the same by decimation; however, it may be necessary to specify the decimation filters to use in this case.

The number of input values for parameters specified as arrays (such as dispersion curves) is limited to 100.

Occasionally *maxsurf* will identify spurious arrivals that happen to pass the existence tests or associate more than one event with the same arrival. These misassociated or spurious arrivals can be identified and removed with post-processing queries.

MsConflict

MsConflict removes spurious or misassociated arrivals from the database and then runs *EvLoc* to obtain the final estimate of M_s . *MsConflict* tries to identify and remove two types of errors: 1) arrivals that are associated with more than one event, and 2) arrivals that are spurious and by chance pass the dispersion test. *MsConflict* runs four queries described in “Identifying and Removing Spurious and Misassociated Arrivals” on page 19.

Input/Processing/Output

MsConflict uses the *getpar.pl* Perl library to provide a standard par interface.

For simplicity, most of the pars are common to the *MsOrid*, *MsConflict*, and *MsInterval* scripts, and therefore, only one par file must be maintained (see “MsInterval” on page 32). Because the par file is shared by three separate scripts, no defaults are supported to ensure that defaults do not differ between scripts.

MsConflict uses the database for input and output. Information is extracted from the **origin**, **assoc**, and **arrival** tables. It updates information (the *qual* code) in the **arrival** table. After completion of spurious arrival tests, it deletes questionable records from the **assoc** and **arrival** tables and changes the corresponding *arid* value in the **amplitude** table to “-1”. The measurement remains in the **parrival** and **amplitude** tables as an upper-bound noise measurement.

▼ Detailed Design

Control

MsConflict is normally run once per day to remove spurious arrivals. Because it uses multiple events in this analysis it must be run outside the normal processing stream, which processes one event at a time. It may be started manually or as part of a daily script. The required input is a Julian date and the database account.

Interfaces

MsConflict is started from a command line. It interacts with the ORACLE database through SQL select, update, and delete queries.

Error States

MsConflict performs a limited amount of error checking. It will fail if the database is not accessible. The logging support offered by the three scripts for surface wave processing (*MsInterval*, *MsOrid*, and *MsConflict*) offers nearly complete control over the messages that will be written to a log file (see “*MsInterval*” on page 32).

libLP

libLP performs great circle path ray-tracing through a regionalized earth model. It is used primarily to return an integrated group velocity between a source and receiver point. This is done for each frequency.

Input/Processing/Output

libLP reads a regionalized earth model (see discussion of “LPcompile” on page 28). All calls to this library are encapsulated within the Pathcor object described under *libpathcor*. The Pathcor object is created and passed the location of the directory containing the regionalized dispersion model. Source and receiver locations and a set of frequencies are then passed to the Pathcor object, and an object method is called to return the integrated group velocities that are calculated through calls to *libLP*.

Control

libLP is started by the Pathcor object, which is created by the *maxsurf* main program.

Interfaces

The interface to *libLP* is hidden within the Pathcor object. Pathcor calls the *libLP* function *LP_trace_rays* to calculate the integrated group velocity.

Error States

libLP will fail and return an error code if the regionalized dispersion curve file is missing or corrupted. There are a few anomalous cases corresponding to grazing incidence along grid boundaries where *libLP* ray-tracing will not find a solution. In this case it returns an error code, and Pathcor will shift the source location slightly and run *libLP* again.

liblpararray

liblpararray beams broadband and long-period array data and searches the beams to find the optimum azimuth and slowness. All array beaming functions are encapsulated within the LPAArray object.

Input/Processing/Output

The LPAArray object requires the location of all stations in the array. One or more data sets corresponding to those locations may be passed to the LPAArray object. LPAArray does not directly access the database or any files, but the array locations, origin, and waveform parameters are extracted from the database by *maxsurf* and then passed to LPAArray. Beaming is performed either by searching for the optimum azimuth and slowness and then beaming on that azimuth and slowness or by beaming on an input azimuth and slowness. The input slowness is specified as a

▼ Detailed Design

phase velocity and may be either a constant (frequency-independent) value, or a frequency-dependent array corresponding to the regionalized phase velocity at the array.

Control

The LPAarray object is created and all related processing is performed within the *proc_array* function that is invoked by the *maxsurf* main program.

Interfaces

The interface to *liblpparray* is through the creation of an LPAarray object. This object maintains its own internal data and is created and accessed through a set of object methods. Internal data and functions are hidden.

Error States

A common error condition is a missing or malfunctioning channel. *liblpparray* identifies these channels by finding the median amplitude and removing channels that are much different from the median. Processing then proceeds with the reduced array. *liblpparray* will fail and return an error code if there are no data or if there is inconsistent input such as a malfunctioning channel.

liblppar

liblppar encapsulates all of the input, output, and database access routines used by *maxsurf*.

Input/Processing/Output

Liblppar works primarily with two data structures: a *Wfmparameter* structure returned by the system library *libwfm* and an *LPparams* structure created by the function *read_lppar*. The *Wfmparameter* structure contains the waveform information normally required for data access. The *LPparams* structure contains all of the additional information required for surface wave processing. Both of these struc-

tures are created from information contained in the par file and any overriding command line arguments using the standard par file syntax. All database access is also handled by functions in *liblppar*. The information retrieved from the database and the information to be written to the database is all stored in the LPparams structure. The LPparams structure and its associated methods can therefore be considered an object encapsulating the data access functions, except that the waveform data accessed through *libwfm* are maintained separately.

Liblppar functions are used within *maxsurf* in the following sequence:

1. The Wfmparameter structure is created by a call to the *libwfm* function *read_libwfm_par*.
2. The LPparams structure is created by a call to the *liblppar* function *read_lppar*.
3. The connection to the database is established and other initialization performed through a call to the *liblppar* function *lplnit*.
4. Origin information is recovered from the database using the input origin ID and the *liblppar* function *find_origin*.
5. Network, station, and array information is recovered from the database. The processing and database queries depend on whether the data to be processed has been specified on input as a network, list of stations, or an array.
6. Old results in the database corresponding to the same origin and stations are deleted.
7. Waveform information is initialized using input parameters.
8. A loop over stations (including sets of stations for an array) that calls *liblppar* routines to get station, channel, and instrument information is performed.
9. After waveforms are retrieved and processed, the results are written to the database using the *liblppar* function *write_results*.
10. After all stations are processed, the database is closed.

▼ Detailed Design

Control

Liblppar is a static library linked with *maxsurf*. It is used by calling its functions. All calls to *liblppar* are made from within the *maxsurf* main program or from within *liblppar* itself.

Interfaces

The interface to *liblppar* is through the creation of the LPparams and Wfmparameter data structures and calling the *liblppar* functions that operate on these data structures.

Error States

Because *liblppar* handles all input and output it can fail in many ways through incorrect or missing data and input values. Considerable effort has been made to ensure that failures do not have catastrophic effects. For example, if information essential for processing is skipped, the station will be skipped and processing will proceed with the next station. The robustness of the library has been tested by running *maxsurf* on continuous data at the PIDC so that it encounters every type of missing data or error condition that is likely to occur in the operational system.

libmag

libmag calculates surface wave magnitudes using a choice of several magnitude formulas. It also identifies the particular waveform segment within the wave train that will be measured, measures it, and saves the amplitude and period to be written to the database. All *libmag* functions are encapsulated within the MagData object.

Input/Processing/Output

libmag requires waveform data, source to receiver distance, and the response of the recording instrument. Additional parameters may be needed for the selected magnitude type. The magnitude type may be one of the following values:

MAGMsPrague, MAGMsVonSeggern, MAGMsTheoretical, MAGMsGutenberg, MAGMsMarshallBasham, MAGMsRP. These values correspond to the Prague (IASPEI), Von Seggern, Theoretical, Gutenberg, Marshall Basham, and Rezapour and Pearce formulas, respectively. The Theoretical formula is based on the equation for propagation of a surface wave, and the Rezapour and Pearce formula uses the Theoretical formula with the optimum values of parameters based on a study of a very large International Seismic Centre (ISC) data set [Rez98]. The current system uses the Rezapour and Pearce formula for all magnitudes.

libmag finds the arrival to be measured by looking through the wavetrain for the largest instrument-corrected arrival within a specified period range and time window. The period range is an input quantity normally set to 18 to 22 seconds. The time window is defined by an input window around the group velocity arrival time. The time limits (relative to origin time) are found using the formulas:

$$t_{\min} = \frac{\text{distance}}{(v_{\text{disp}} + v_{\text{displimit}})} - (p_{\text{displimit}} \cdot \text{period}) - t_{\text{displimit}}$$

(1)

and

$$t_{\max} = \frac{\text{distance}}{(v_{\text{disp}} - v_{\text{displimit}})} + (p_{\text{displimit}} \cdot \text{period}) + t_{\text{displimit}}$$

where *vdisp* is the estimated group velocity at the measured period, *vdisplimit* is an input value specifying the allowable group velocity range, and *pdisplimit* and *tdisplimit* are adjustments to the time limits that allow the time window to increase with period and allow for a minimum time window. The PIDC uses values of *vdisplimit*=0.2, *pdisplimit*=1.0, and *tdisplimit*=0.

Control

libmag is a static library linked with *maxsurf*. The *MagData* object is created by the *proc_mag* function that is invoked by the *maxsurf* main program.

▼ Detailed Design

Interfaces

The input to *libmag* is through the creation of a *MagData* object. This object maintains its own internal data and is created and accessed through a set of object methods. Internal data and functions are hidden.

Error States

libmag will return zero amplitude and zero magnitude if it is unable to find an arrival within the specified period band and time window.

libnbf

libnbf performs narrow-band filtering of waveforms and compares the measured arrival times with a predicted group velocity curve. All narrow-band filtering operations are encapsulated within the *NbfData* object.

Input/Processing/Output

The *NbfData* object requires the waveform and waveform information, instrument response, and frequencies to be used for narrow-band filtering. If the dispersion test will be performed, it also requires an input dispersion curve (usually generated from a regionalized dispersion model) covering the range of frequencies. Other properties of *NbfData* that may be set are the filter Q and the minimum amplitude that should be considered an arrival. *NbfData* finds the arrival time of the largest amplitude arrival and the corresponding instantaneous frequency. In general, the output frequency differs from the input frequency because the finite bandwidth of the filter applied to a frequency-dependent spectrum causes the maximum amplitude to occur close to but not exactly on the input frequency.

The surface wave existence test compares the measured group velocity arrival time at the measured instantaneous frequency with the predicted group velocity for the path at the same frequency. The arrival must fall within a time window defined by equations (1). For many waveforms, the measured values at some frequencies will fall outside of this range, particularly near the high or low frequency limits where

signal-to-noise ratio may be low, or at frequencies where there is interference due to multipathing. Therefore, the detection test requires that only a certain fraction of the arrivals fall within these bounds. This number should be set high enough to minimize spurious arrivals, but low enough that typical surface wave data will pass the test. The PIDC uses input frequencies of 0.02, 0.025, 0.03, 0.035, 0.04, 0.45, 0.05, 0.06 Hz, and requires that at least 70% of the arrivals (6 out of 8) pass the test.

Control

The NbfData object is created and all related processing is performed within the *proc_nbf* function that is invoked by the *maxsurf* program.

Interfaces

The interface to *libnbf* is through the creation of an NbfData object. This object maintains its own internal data and is created and accessed through a set of object methods. Internal data and functions are hidden.

Error States

libnbf is quite robust. However, it can produce some anomalous results when applied to real data. The filter width is defined by a filter Q so that the filter width is narrower at higher frequencies. The filter Q is normally set to 15, which is a good worldwide average value. However, for optimal performance the filter Q should be set to a smaller value at close range where better time resolution is needed and a larger value at larger distances. *libnbf* may fail to find a dispersion curve if there are interfering arrivals or strong interference between signal and noise.

libpathcor

libpathcor creates surface wave path corrections and is primarily used to construct phase and group velocity dispersion curves for any source and receiver point. All path correction operations are encapsulated within the Pathcor object.

▼ Detailed Design

Input/Processing/Output

The Pathcor object encapsulates all path correction operations and all calls to the *libLP* library, which integrates dispersion curves over a great circle path. The Pathcor object is usually passed the directory location and filenames of the regionalized group and phase velocity models. Path dependent dispersion curves are created by setting the desired set of frequencies and passing the object the source and receiver coordinates.

Control

The Pathcor object is created by the *maxsurf* main program, and its methods are called by the *maxsurf* main program to calculate a regionalized group velocity curve, and optionally by the *proc_array* function to calculate the phase velocity at the receiving station for performing regionalized beaming.

Interfaces

The interface to *libpathcor* is through creation of a Pathcor object. This object maintains its own internal data and is created and accessed through a set of object methods. Internal data and functions are hidden.

Error States

libpathcor will fail if the regionalized dispersion curve files do not exist or are not in the specified location. For certain grazing paths, *libLP* will fail. In that case, *libpathcor* shifts the location slightly and recalculates the dispersion curves. It will shift twice, once to the east and once to the north before failing.

libpmotion

libpmotion performs polarization filtering to determine particle motion and estimate the back azimuth to the event origin. All polarization filtering operations are encapsulated within the PolFilter object.

Input/Processing/Output

The PolFilter object encapsulates all polarization filtering operations. After it is created, it is passed 3-C data transformed to the frequency domain with the instrument response removed. A frequency band is then specified, and the estimated azimuth, F-statistic, amplitude, and ellipticity are returned for this frequency band.

Control

The PolFilter object is created by the *proc_poltest* function, which is invoked by the *maxsurf* main program.

Interfaces

The interface to *libpmotion* is through the creation of a PolFilter object. The object maintains its own internal data and is created and accessed through a set of object methods. Internal data and functions are hidden.

Error States

Errors will result if the component gains and responses are not matched properly. *libpmotion* will fail and return an error message if any of the three components of data do not exist or are all zero. *libpmotion* will fail and return an error message if the frequency spacing is undefined or the limits of the requested frequency band are outside the limits of the transformed waveform. *libpmotion* will fail and return an error message if passed an unknown wave type. Allowable wave types are RayLove, Love, SH, P, and SV. Only RayLove is used by *maxsurf*.

libresp

libresp performs instrument response calculations. All operations are encapsulated within the InstData object.

▼ Detailed Design

Input/Processing/Output

The `InstData` object is primarily a wrapper around the IDC *libresponse* library and calls the library routines *scaled_response* or *unscaled_response* to calculate the actual instrument responses. Input to the `InstResp` object consists of the directory and file containing the response, the calibration period, and the calibration factor. `InstResp` then returns the instrument response in a variety of formats for an input set of frequencies. The frequencies may be specified as an evenly spaced array or a discrete set of frequencies.

Control

The heavily used `InstData` object is created by the *proc_array*, *proc_vertcomp*, and *proc_poltest* functions, which are invoked by the *maxsurf* main program. The `InstData` object created by *proc_array* (for array data) or *proc_vertcomp* (for vertical component data) is returned to the *maxsurf* main program and then passed to the *proc_nbf* and *proc_mag* functions for narrow-band filtering and estimating magnitudes. *Proc_array* and *proc_vertcomp* both replace the instrument response with a standard (usually KS36000 LP) instrument, and the `InstData` object returned corresponds to the new instrument response.

Interfaces

The interface to *libresp* is through the creation of an `InstData` object. This object maintains its own internal data and is created and accessed through a set of object methods. Internal data and functions are hidden.

Error States

Because *libresp* calls functions in *libresponse*, it transmits errors in *libresponse* functions. There is currently an error evaluating Infinite Impulse Response (IIR) filters in *libresponse*. *libresp* will fail if either the directory or the filename of the response file is incorrect.

libs3obj

libs3obj manages waveform data and is primarily a wrapper around the *libwfm* library. *libs3obj* operations are encapsulated within the WfmSeisData object. WfmSeisData manages sets of seismograms. Data for a single seismogram are managed by the SeisData object.

Input/Processing/Output

The WfmSeisData object is initialized by passing a pointer to the Wfmparameter structure, which is used by *libwfm* and created from the input par file. The WfmSeisData object has methods to perform a variety of operations and then retrieve data from the file system. WfmSeisData calls *libwfm* routines to retrieve data. *libwfm* uses the ORACLE database to retrieve information about the waveform from the database and to retrieve the data from the file system. WfmSeisData data operations include decimation, demean, detrend, rotate, and Fourier transform. WfmSeisData can also be passed origin time and distance so that data can be retrieved by group velocity interval. WfmSeisData can also retrieve instrument calibration period and calibration factor. This is necessary because the **sensor** table is not maintained by the IDC so the only accurate calibration information is contained in the **wfdisc** table.

Control

The WfmSeisData object is created by the *maxsurf* main program and provides the interface to the waveform data. It is passed to the *proc_array*, *proc_vertcomp*, and *proc_poltest* functions, which are invoked by the *maxsurf* main program.

Interfaces

The interface to *libs3obj* is through the creation of a WfmSeisData object. This object maintains its own internal data and is created and accessed through a set of methods. Internal data and functions are hidden.

▼ Detailed Design

Error States

Errors due to data requests for unavailable data are very common. *WfmSeisData* returns NULL in response to a data request if errors occur.

libutilc

libutilc contains utility functions written in the C programming language. Functions used by *maxsurf* include various time operations.

Input/Processing/Output

libutilc functions are used to calculate the Julian date from epoch time and to calculate time intervals to assess performance of various parts of *maxsurf* processing. It is also used to create the *lddate* string from the current time.

Control

libutilc functions are called by the *maxsurf* main program and by *liblppar* routines.

Interfaces

libutilc is a simple function library and the interface is through calls to its functions.

libutilf

libutilf contains a set of utility functions written in the FORTRAN programming language. These are standard numerical routines for performing Fourier transforms, interpolating, statistical operations, and operations with complex numbers.

Input/Processing/Output

Fourier transform requires an input time series, number of points, and sample rate. Interpolation requires input independent and dependent arrays and number of points. Most operations with complex numbers are performed with FORTRAN routines because of FORTRAN's native support for complex arithmetic.

Control

libutilf functions are called from many parts of *maxsurf* whenever numerical processing or complex arithmetic is required.

Interfaces

libutilf is a simple library of routines. It is invoked by calling its functions.

Error States

libutilf will fail if the numerical routines are passed improper input such as negative sampling rates or a negative number of data points. Most routines require specification of the number of points passed as arrays. The array size must be as long as the number of points or the routine could fail with unpredictable side effects. *libutilf* routines do not protect against improper inputs, so calling routines must check the parameters that are passed to *libutilf*.

DATABASE DESCRIPTION

The SWIM software uses the database to retrieve information about event origin, station location, instrument orientation, instrument response, and waveforms, and to write information about surface wave arrival times and measurements for use by *EvLoc* in estimating surface wave magnitudes. The SWIM software interacts with the database through the Generic Database Interface (GDI).

Database Design

The entity-relationship model of the schema is indicated in Figure 5. *maxsurf* extracts event information from the **origin** table, station and instrument information from the **site**, **sitechan**, **sensor**, and **instrument** tables, and waveform information from the **wfdisc** table. Results are written by *maxsurf* to the **arrival**, **assoc**, **parival**, and **amplitude** tables. *maxsurf* does not write to the **stamag** or **netmag** tables, but a postprocessor, *EvLoc*, uses information written by *maxsurf* to create the information in these two tables.

▼ Detailed Design

Database Schema

Table 3 shows the usage of database tables by the SWIM software. For each table used, the third column shows the purpose for reading or writing each attribute. The relationships between the tables are shown in the entity-relationship diagrams of Figure 5.

TABLE 3: DATABASE USAGE BY SWIM SOFTWARE

Table	Action	Attribute Usage
affiliation	read	<i>MsOrid</i> reads <i>net</i> and <i>sta</i> to determine a channel list to send to <i>maxsurf</i> .
amplitude	write, delete	<p><i>MsOrid</i> deletes rows based on assoc.arid to remove old LRs.</p> <p><i>MsOrid</i> deletes rows based on parrival.parid to remove old LRs.</p> <p>For each surface wave measurement, whether LR or noise, <i>maxsurf</i> writes <i>ampid</i>, <i>arid</i> or <i>parid</i> (for LR or noise, respectively), <i>chan</i> (1z or bz), <i>amp</i>, <i>per</i>, <i>amptime</i>, <i>start_time</i>, <i>duration</i>, <i>amptype</i> (ALR/2 for LR, ANL/s for noise), <i>units</i> (nm), and <i>auth</i> (version of <i>maxsurf</i>).</p> <p><i>MsConflict</i> sets <i>arid</i>=-1 and <i>inarrival</i>= 'n' for <i>arids</i> based on arrival.qual, assoc.arid, assoc.phase, assoc.orid, origin.orid, and origin.time.</p>
arrival	read, write, delete	<p><i>MsOrid</i> deletes rows based on assoc.arid to remove old LRs.</p> <p><i>MsConflict</i> updates <i>qual</i> based on <i>arid</i>, <i>sta</i>, <i>time</i>, <i>iphase</i>, assoc.arid, assoc.delta, assoc.phase, assoc.orid, origin.orid, and origin.time.</p> <p><i>MsConflict</i> deletes rows based on <i>qual</i>, assoc.arid, assoc.phase, assoc.orid, origin.orid, and origin.time.</p>

TABLE 3: DATABASE USAGE BY SWIM SOFTWARE (CONTINUED)

Table	Action	Attribute Usage
assoc	read, write, delete	<p><i>MsOrid</i> uses <i>phase</i> and <i>orid</i> to determine <i>arid</i> rows that will be deleted from amplitude and arrival.</p> <p><i>MsOrid</i> uses <i>phase</i> and <i>orid</i> to remove old LR rows.</p> <p><i>MsOrid</i> reads <i>sta</i>, <i>orid</i>, and <i>phase</i> to determine a channel list to send to <i>maxsurf</i> and to determine the station list to pass to <i>EvLoc</i>.</p> <p><i>maxsurf</i> writes rows.</p> <p><i>MsConflict</i> reads <i>arid</i>, <i>delta</i>, <i>phase</i>, and <i>orid</i> to use when updating arrival.qual.</p> <p><i>MsConflict</i> deletes rows based on <i>arid</i>, <i>phase</i>, <i>orid</i>, arrival.qual, origin.orid, and origin.time.</p> <p><i>MsConflict</i> reads <i>sta</i>, <i>orid</i>, and <i>phase</i> to determine the station list to pass to <i>EvLoc</i>.</p>
instrument	read	<i>maxsurf</i> reads <i>dir</i> and <i>dfile</i> to get the instrument response file.
netmag	read, write, delete	<p><i>MsOrid</i> deletes rows based on stamag.magid to remove old LRs.</p> <p><i>EvLoc</i> writes <i>magid</i>, <i>net</i>, <i>magtype</i> (ms_ave and others), <i>nsta</i>, <i>magnitude</i>, <i>uncertainty</i>, and <i>auth</i>.</p> <p><i>MsConflict</i> reads <i>magid</i>, <i>magnitude</i>, and <i>magtype</i> for use in updating origin.msid, origin.ms, and origin.nass.</p>

▼ Detailed Design

TABLE 3: DATABASE USAGE BY SWIM SOFTWARE (CONTINUED)

Table	Action	Attribute Usage
origin	read, write	<p><i>MsInterval</i> reads <i>orid</i> and <i>time</i> for identifying origins in an interval.</p> <p><i>MsOrid</i> sets <i>ms</i>=-999 and <i>msid</i>=-1 for the working <i>orid</i>.</p> <p><i>MsOrid</i> reads <i>orid</i>, <i>mb</i>, and <i>depth</i> to screen origins sent to <i>maxsurf</i>.</p> <p><i>maxsurf</i> reads:</p> <ul style="list-style-type: none"> • <i>orid</i> and <i>evid</i> for event identification • <i>lat</i>, <i>lon</i>, and <i>depth</i> for computing travel times and station ranking using <i>libloc</i> functions • <i>time</i> for computing intervals • <i>jdate</i> for constraint in querying site and sitechan • <i>mb</i> for computing detection probability <p><i>EvLoc</i> updates <i>msid</i> and <i>ms</i>.</p> <p><i>MsOrid</i> updates <i>nass</i> for the working <i>orid</i>.</p> <p><i>MsConflict</i> reads <i>time</i> and <i>orid</i> to use when updating arrival.qual, to determine <i>arid</i> rows to be removed from arrival and assoc, and to determine <i>arid</i> rows to be updated in amplitude.</p> <p><i>MsConflict</i> reads <i>orid</i> and <i>time</i> to determine the station list to pass to <i>EvLoc</i>.</p> <p><i>MsConflict</i> reads <i>orid</i>, <i>nass</i>, <i>ndef</i>, <i>time</i>, and the netmag table and updates <i>msid</i>, <i>ms</i>, and <i>nass</i>.</p>
origerr	read	<p><i>MsOrid</i> reads <i>sdepth</i> to use in a test of depth for origins sent to <i>maxsurf</i>.</p>
parrival	read, write, delete	<p><i>MsOrid</i> uses <i>phase</i> and <i>orid</i> to determine <i>parid</i> rows that will be deleted from amplitude.</p> <p><i>MsOrid</i> uses <i>phase</i> and <i>orid</i> to remove old LR rows.</p> <p>For each surface wave measurement, whether LR or noise, <i>maxsurf</i> writes <i>parid</i>, <i>orid</i>, <i>evid</i>, <i>sta</i>, <i>time</i>, <i>phase</i> (LR), and <i>delta</i>.</p> <p><i>MsOrid</i> reads <i>sta</i>, <i>orid</i>, and <i>phase</i> to determine the station list to pass to <i>EvLoc</i>.</p> <p><i>MsOrid</i> uses <i>sta</i>, <i>orid</i>, and <i>phase</i> to delete the <i>parid</i> rows that are duplicates.</p>

TABLE 3: DATABASE USAGE BY SWIM SOFTWARE (CONTINUED)

Table	Action	Attribute Usage
sensor	read	<i>MsOrid</i> reads <i>sta</i> and <i>chan</i> to determine a channel list to send to <i>maxsurf</i> . <i>maxsurf</i> reads <i>calratio</i> and <i>calper</i> to get instrument response information.
site	read	<i>maxsurf</i> reads <i>lat</i> and <i>lon</i> to get site location information.
sitechan	read	<i>maxsurf</i> reads <i>hang</i> and <i>vang</i> to get instrument orientation.
stamag	read, write, delete	<i>MsOrid</i> uses <i>phase</i> and <i>orid</i> to determine <i>magid</i> rows that will be deleted from netmag . <i>MsOrid</i> uses <i>phase</i> and <i>orid</i> to remove old LR rows. <i>EvLoc</i> writes <i>magid</i> , <i>ampid</i> , <i>sta</i> , <i>arid</i> , <i>orid</i> , <i>evid</i> , <i>delta</i> , <i>phase</i> (LR), <i>magtype</i> (<i>ms_ave</i> and others), <i>magnitude</i> , <i>mmodel</i> (<i>rez_pearce</i>), and <i>auth</i> .
wfdisc	read	<i>maxsurf</i> reads <i>chan</i> , <i>time</i> , <i>endtime</i> , <i>nsamp</i> , <i>samprate</i> , <i>dir</i> , and <i>dfile</i> to get waveform information.

▼ Detailed Design

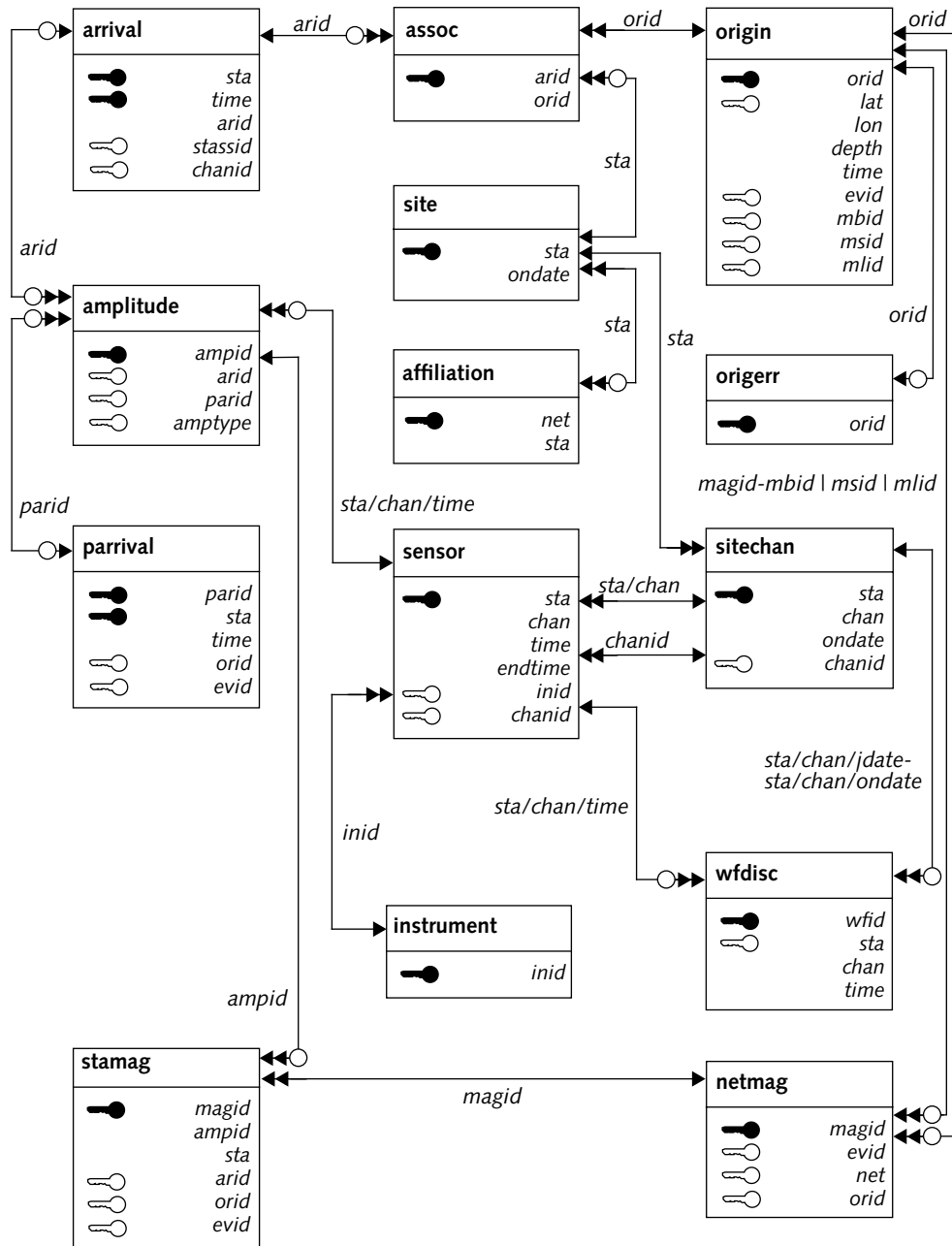


FIGURE 5. TABLE RELATIONSHIPS FOR SWIM SOFTWARE

References

The following sources supplement or are referenced in this document:

- [DOD94a] Department of Defense, "Software Design Description," *Military Standard Software Development and Documentation*, MIL-STD-498, 1994.
- [IDC5.1.1Rev2] Science Applications International Corporation, Veridian Pacific-Sierra Research, *Database Schema, Revision 2*, SAIC-00/3057, PSR-00/TN2830, 2000.
- [IDC5.2.1] Science Applications International Corporation, *IDC Processing of Seismic, Hydroacoustic, and Infrasonic Data*, SAIC-99/3023, 1999.
- [IDC6.5.2Rev0.1] Science Applications International Corporation, *Distributed Application Control System (DACS) Software User Manual, Revision 0.1*, SAIC-00/3038, 2000.
- [Mar72] Marshall, P. D., and Basham, P. W., "Discriminating Between Earthquakes and Underground Explosions Employing an Improved Ms Scale," *Geophysical Journal of the Royal Astronomical Society*, Volume 28, pp. 431–458, 1972.
- [Nor94] North, R. G., and Woodgold, C. R. D., "Automated Detection and Association of Surface Waves," *Annali de Geofisica*, Volume 37, pp. 301–308, 1994.

▼ References

- [Rez98] Rezapour, M., and Pearce, R. G., "Bias in Surface-wave Magnitude M_s Due to Inadequate Distance Correction," *Bulletin of the Seismological Society of America*, Volume 88, pp. 43–61, 1998.
- [Sma71] Smart, E., and Flinn, E. A., "Frequency-Wave Number Analysis and Fisher Signal Detection in Real-Time Infrasonic Array Data Processing," *Geophysical Journal of the Royal Astronomical Society*, Volume 26, pp. 279–284, 1971.
- [Ste86] Stevens, J. L., "Estimation of Scalar Moments from Explosion-Generated Surface Waves," *Bulletin of the Seismological Society of America*, Volume 76, pp. 123-151, 1986.
- [Ste96] Stevens, J. L., *Regionalized Maximum Likelihood Surface Wave Analysis*, Maxwell Technologies Technical Report, PL-TR-96-2273, SSS-DTR-96-15562, ADA 321813, 1996.
- [Ste97] Stevens, J. L., and McLaughlin, K. L., *Improved Methods for Regionalized Surface Wave Analysis*, Phillips Laboratory, Directorate of Geophysics, AFMC, Hanscom AFB, MA, PL-TR-97-2135, 1997.
- [Ste01] Stevens, J. L., and McLaughlin, K. L., "Optimization of Surface Wave Identification and Measurement," *Pure and Applied Geophysics*, Volume 158, 2001.

Glossary

Symbols

3-C

Three-component.

A

amplitude

Zero-to-peak height of a waveform in nanometers.

array

Collection of sensors distributed over a finite area (usually in a cross or concentric pattern) and referred to as a single station.

arrival

Signal that has been associated to an event. First, the Global Association (GA) software associates the signal to an event. Later during interactive processing, many arrivals are confirmed and improved by visual inspection.

ASCII

American Standard Code for Information Interchange. Standard, unformatted 256-character set of letters and numbers.

associated phase

Phase that is associated with an S/H/I event.

azimuth

Direction, in degrees clockwise with respect to North, from a station to an event.

B

background noise

Natural movements of the earth, oceans, and atmosphere as seen on S/H/I sensors (usually measured in data preceding signals).

beam

Waveform created from array station elements that are sequentially summed in the direction of a specified azimuth and slowness.

bulletin

Chronological listing of event origins spanning an interval of time. Often, the specification of each origin or event is accompanied by the event's arrivals and sometimes with the event's waveforms.

▼ Glossary

C**channel**

Component of motion or distinct stream of data.

child process

UNIX process created by the *fork* routine. The child process is a snapshot of the parent at the time it called *fork*.

CMR

Center for Monitoring Research.

command

Expression that can be input to a computer system to initiate an action or affect the execution of a computer program.

component

(1) One dimension of a three-dimensional signal; (2) The vertically or horizontally oriented (north or east) sensor of a station used to measure the dimension; (3) One of the parts of a system; also referred to as a module or unit.

Comprehensive Nuclear-Test-Ban Treaty Organization

Treaty User group that consists of the Conference of States Parties (CSP), the Executive Council, and the Technical Secretariat.

Computer Software Component

Functionally or logically distinct part of a computer software configuration item, typically an aggregate of two or more software units.

Computer Software Configuration Item

Aggregation of software that is designed for configuration management and treated as a single entity in the configuration management process.

Conference of States Parties

Principal body of the CTBTO consisting of one representative from each State Party accompanied by alternate representatives and advisers. The CSP is responsible for implementing, executing, and verifying compliance with the Treaty.

COTS

Commercial-Off-the-Shelf; terminology that designates products such as hardware or software that can be acquired from existing inventory and used without modification.

CSC

Computer Software Component.

CSCI

Computer Software Configuration Item.

CTBT

Comprehensive Nuclear-Test-Ban Treaty (the Treaty).

CTBTO

Comprehensive Nuclear-Test-Ban Treaty Organization; Treaty User group that consists of the Conference of States Parties (CSP), the Executive Council, and the Technical Secretariat.

D**DACS**

Distributed Application Control System. This software supports inter-application message passing and process management.

data flow

Sequence in which data are transferred, used, and transformed during the execution of a computer program.

DBMS

Database Management System.

detailed design

Refined and expanded version of the preliminary design of a system or component. This design is complete enough to be implemented.

dispersion

Expansion of the length of a seismic wavetrain due to each wavelength travelling with its own velocity.

E**ellipticity**

Ratio of horizontal to vertical motion of seismic waves.

entity-relationship (E-R) diagram

Diagram that depicts a set of entities and the logical relationships among them.

epoch time

Number of seconds after January 1, 1970 00:00:00.0.

event

Unique source of seismic, hydroacoustic, or infrasonic wave energy that is limited in both time and space.

F**features**

Various measurements of a transient signal used to characterize an arrival.

FFT

Fast Fourier Transform.

filesystem

Named structure containing files in sub-directories. For example, UNIX can support many filesystems; each has a unique name and can be attached (or mounted) anywhere in the existing file structure.

fork

UNIX system routine that is used by a parent process to create a child process.

G**GB**

Gigabyte. A measure of computer memory or disk space that is equal to 1,024 megabytes.

GDI

Generic Database Interface.

group velocity

Propagation velocity of packets or groups of waves.

▼ Glossary

H**hydroacoustic**

Pertaining to sound in the ocean.

Hz

Hertz.

I**IASPEI**

International Association of Seismology and Physics of the Earth's Interior.

ID

Identification; identifier.

IDC

International Data Centre.

IIR

Infinite Impulse Response (filters also referred to as recursive filters).

IMS

International Monitoring System.

IPC

Interprocess communication. The messaging system by which applications communicate with each other through *libipc* common library functions. See *tuxshell*.

ISC

International Seismological Centre.

J**Julian date**

Increasing count of the number of days since an arbitrary starting date.

K**KB**

Kilobyte. 1,024 bytes.

km

Kilometer.

L**LAN**

Local Area Network.

LP

Long period.

LR

Rayleigh wave. A seismic phase that travels along the surface of the earth.

M**MB**

Megabyte. 1,024 kilobytes.

 m_b

Magnitude of a seismic body wave.

 M_s

Magnitude of seismic surface waves.

N

noise

Incoherent natural or artificial perturbations of the waveform trace caused by ice, animals migrations, cultural activity, equipment malfunctions or interruption of satellite communication, or ambient background movements.

O

ORACLE

Vendor of the database management system used at the PIDC and IDC.

orid

Origin Identifier.

origin

Hypothesized time and location of a seismic, hydroacoustic, or infrasonic event. Any event may have many origins. Characteristics such as magnitudes and error estimates may be associated with an origin.

P

par

See parameter.

parameter

User-specified token that controls some aspect of an application (for example, database name, threshold value). Most parameters are specified using [*token* = *value*] strings, for example, `dbname=mydata/base@oracle`.

parameter (par) file

ASCII file containing values for parameters of a program. Par files are used to replace command line arguments. The files are formatted as a list of [*token* = *value*] strings.

parent process

UNIX process that creates a child process by the *fork* routine. The child process is a snapshot of the parent at the time it called *fork*.

period

Average duration of one cycle of a phase, in seconds per cycle.

phase

Arrival that is identified based on its path through the earth.

phase name

Name assigned to a seismic, hydroacoustic or infrasonic arrival associated with a travel path.

PIDC

Prototype International Data Centre.

polarization

Form of three-component analysis used to derive azimuth and slowness information from non-array stations.

probability of detection

Probability estimate that an arrival from a given event will be detected at a station given the location and magnitude of the event, the average noise level and its standard deviation at the station, and the signal-to-noise detection threshold.

▼ Glossary

Q**Q**

Dimensionless quality factor inversely related to the strength of attenuation of seismic wave amplitudes.

QC

Quality Control.

R**RDBMS**

Relational Database Management System.

REB

Reviewed Event Bulletin; the bulletin formed of all S/H/I events that have passed analyst inspection and quality assurance review.

run

(1) Single, usually continuous, execution of a computer program. (2) To execute a computer program.

S**S/H/I**

Seismic, hydroacoustic, and infrasonic.

slowness

Inverse of velocity, in seconds/degree; a large slowness has a low velocity.

Solaris

Name of the operating system used on Sun Microsystems hardware.

SQL

Structured Query Language; a language for manipulating data in a relational database.

States Parties

Treaty user group who will operate their own or cooperative facilities, which may be NDCs.

station

Collection of one or more monitoring instruments. Stations can have either one sensor location (for example, BGCA) or a spatially distributed array of sensors (for example, ASAR).

surface wave

Seismic wave propagating along the surface of the earth.

T**time, epoch**

See epoch time.

Treaty

Comprehensive Nuclear-Test-Ban Treaty (CTBT).

tuxshell

Process used to execute applications lacking interprocess communications capabilities through Tuxedo. See IPC.

U

UNIX

Trade name of the operating system used by the Sun workstations.

USGS

United States Geological Survey.

W

waveform

Time-domain signal data from a sensor (the voltage output) where the voltage has been converted to a digital count (which is monotonic with the amplitude of the stimulus to which the sensor responds).

wfdisc

Waveform description record or table.

workstation

High-end, powerful desktop computer preferred for graphics and usually networked.

Index

A

affiliation 15, 76, 80
amplitude 9, 15, 21, 36, 61, 76, 80
arrival 9, 15, 20, 21, 36, 61, 76, 80
assoc 15, 20, 21, 36, 61, 77, 80
Atmospheric Transport CSC 3

B

beam 25
 forming from array data 63

C

COTS software 10

D

DACS 12
database usage 76
data flow 24
 LPcompile 24
 maxsurf 24
 MsInterval 24
 MsOrid 24
 symbols iv
development status 8

disk space required 9

E

entity-relationship
 diagram 80
 symbols v
Event Screening CSC 3
EvLoc 5, 7, 8, 12, 19, 36
 database use 75
 logging 34
 monitoring through *MsOrid* 37

G

gridfile ASCII format 28

H

hardware 9
 configuration 10
history 8

I

instrument 15, 77, 80
instrument response calculations 71
interval 21

K

KS36000 25

▼ Index

L

libLP
detailed design 62
liblpararray
detailed design 63
liblppar
detailed design 64
libmag
detailed design 66
libnbf
detailed design 68
libpathcor
detailed design 69
libpmotion
detailed design 70
libraries 13
libresp
detailed design 71
libutilc
detailed design 74
libutilf
detailed design 74
LPcompile 16, 18
data flow 24
detailed design 28

M

magnitudes 19
maxsurf 8, 16
database use 75
data flow 24
detailed design 37
example par file 57
logging 34
parameters 38
memory required 9
MsConflict 12, 16
detailed design 61
parameters 32
queries 19

M_s formula 19
MsInterval 12, 16, 18
data flow 24
detailed design 32
parameters 32
MsOrid 12, 16, 18
data flow 24
detailed design 36
parameters 32

N

netmag 15, 36, 77, 80
Network Processing CSC 2

O

Operational Scripts CSC 3
origerr 15, 78, 80
origin 15, 18, 37, 61, 78, 80

P

parameter file
maxsurf 38
parrival 15, 21, 36, 61, 78, 80
path corrections 69
polarization analysis 26
polarization filtering 70
Post-location Processing CSC 2

R

Radionuclide Processing CSC 3
ray-tracing 62
regionalized dispersion curves 28
disk space requirements 9
regionalized dispersion model 68
regionalized earth model 62

S

- sensor** 15, 79, 80
- site** 15, 79, 80
- sitech** 16, 60, 79, 80
- spurious arrivals 62
- stamag** 16, 36, 79, 80
- Station Processing CSC 2
- surface wave
 - existence test 68
 - path corrections 69
 - search 18

T

- Time-series Libraries CSC 3
- Time-series Tools CSC 3
- tuxshell*
 - calling *MsInterval* 32
- typographical conventions v

U

- utility functions 74

V

- velfile ASCII format 30

W

- wfdisc** 16, 79, 80

